UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

EVOLUCIÓN TECNOLÓGICA DE UNA APLICACIÓN EXISTENTE DESARROLLADA EN WINDOWS MOBILE A ANDROID MEDIANTE XAMARIN

Guillermo Carniado Cartas

Tutor: David González López

Ponente: Luis Fernando Lago Fernández

JUNIO_2015

RESUMEN DEL TRABAJO FIN DE GRADO

El siguiente proyecto abarca la evolución tecnológica de una aplicación desarrollada en Windows Mobile. La mencionada evolución consistirá en dotar de una serie de mejoras a la aplicación original y, fundamentalmente, su migración al sistema operativo Android. Este conjunto de cambios supondrá, en consecuencia, un cambio de versión.

El proyecto original consistía en una aplicación de seguimiento de rutas de atención domiciliaria, en la cual cada técnico puede acceder a una ruta determinada con información acerca de las visitas y procedimientos que éste debe realizar. A su vez, la aplicación permite documentar todo el proceso y sincronizarlo con un servidor central.

Las necesidades del cliente requerían la migración de dicha aplicación a un entorno de ejecución Android.

El trabajo presentado abarca tanto la referida migración, como los distintos cambios y mejoras contenidos en el cambio de versión.

Para la realización de la citada migración ha sido necesario completar las siguientes tareas:

- Análisis exhaustivo de la aplicación original.
- Desarrollo de una interfaz de usuario desde cero.
- Comunicación entre el terminal y el back office de la aplicación original.
- Migración de la base de datos SQLServerMobile a SQLite para crear una base de datos local en el terminal.
- Análisis, diseño y desarrollo de las nuevas funcionalidades.
- Realizar pruebas y corregir los errores encontrados previamente a cada entrega.
- Solucionar las disconformidades recibidas mediante la retroalimentación posterior a cada entrega.

De este modo, mis propósitos en el desarrollo de la citada aplicación como TRABAJO DE FIN DE GRADO consistirán en:

- Comprender el funcionamiento y lógica de la aplicación base, así como de su estructura y flujo de datos con el back office.
- Desarrollar un esqueleto gráfico para Android mediante la plataforma Xamarin y el lenguaje de programación C#.
- Migrar la base de datos existente en SQLServerMobile, así como su comunicación con el back office de la aplicación a SQLite.
- Desarrollar las nuevas funcionalidades requeridas por el cliente.

PALABRAS CLAVE

ANDROID, *XAMARIN* y *MULTIPLATAFORMA*.

GLOSARIO

Tabla 1: GLOSARIO	
HMS	Homes
HMCH	Homes Chorus, menciona a la aplicación base y a su back office
DTO	Objeto de transferencia de datos
DAM	Gestor de acceso a datos
CSV	Valores separados por coma
DN	Nota de entrega
eDN	Nota de entrega electrónica (Versión electrónica, en PDF de una DN)
AL	Air Liquide

ABSTRACT

The present Project covers the technological evolution of an application developed in Windows Mobile. The evolution will consist on a collection of improvements of the original application and, above all, in its migration into an Android Operating System. This set of changes will form a new version.

The original project consisted in a home care route tracking app, in which each technician can access a predefined route with information and procedures of the visit. In addition, the application documents all the process and synchronizes with the central server.

Customer needs require the migration of the application to an Android execution environment.

The work covers migration as well as different changes and improvements contained into the new version.

To the implementation of migration it is required to complete the following tasks:

- Exhaustive analysis of the original version.
- Development of user interface from the beginning.
- Communication between terminal and original version back office.
- Migration from SQLServerMobile database to SQLite to create a local database in the terminal.
- Analysis, design and development of new functionalities.
- Test performance and correction of mistakes found previously to the deliver.
- Resolution of disconformities through retro feeding before the deliver.

In that way, the purposes on the development of the named application as end university work are the following:

- Comprehension of functionality and logic of the base application and its structure and dataflow with the back office.
- Development of a graphic structure for Android through Xamarin platform and C# programming language
- Migration of the existing database SQLServerMobile and its communication with the SQLite application back office.
- Development of new functionalities required by the client.

KEYWORDS

ANDROID, XAMARIN y CROSS-PLATFORM.

GLOSSARY

Tabla 2: GLOSSARY	
HMS	Homes
HMCH	Homes Chorus (Previous version and its back office)
DTO	Data Transfer Object
DAM	Data Access Manager
CSV	Comma-separated Values
DN	Delivery Note
eDN	Electronic Delivery Note (Electronic version, in PDF of DN)
AL	Air Liquide

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este apartado se pretende acotar el proyecto, señalando las motivaciones de las que parte, los objetivos que pretende cumplir y presentando la estructura del documento.

1.1. Motivación

La motivación del proyecto surge de la necesidad de adaptar una herramienta consolidada a las nuevas tecnologías existentes.

Tras el auge de Android e iOS y la caída paralela de Windows Mobile, Microsoft decidió descontinuar su sistema operativo para concentrarse en Windows Phone.

Actualmente el soporte de Windows Mobile está finalizado desde agosto de 2013 [(1)]. Esto supone entre otros problemas un encarecimiento y limitación del hardware, por lo que se comprende la decisión de migrar la herramienta a un sistema operativo más moderno. De este modo los fabricantes de terminales industriales son forzados a crear terminales con Android.

Para reemplazarlo Microsoft mantiene a día de hoy el soporte de Windows Embedded Compact 7 y de Windows Embedded Handheld 6.5. Ambas plataformas permiten una migración menos pronunciada que Windows Phone, donde el nuevo enfoque a usuarios finales mediante el Market choca con el anterior, centrado en soluciones empresariales [(2)].

Aunque es cierto que se mantiene el soporte para ambas plataformas, no son compatibles con las últimas versiones de Visual Studio y además, la oferta de terminales compatibles con esta tecnología es menor que la de Android o Windows Phone. De modo que con vistas al futuro esta solución parece perder en la comparación con los sistemas operativos móviles, lo que explicaría su descarte a la hora de seleccionar el sistema operativo.

Esta migración ofrece una oportunidad de añadir distintas funcionalidades para la mejora del sistema, englobando tanto la migración como las funcionalidades añadidas en una evolución de la versión de la herramienta.

1.2. Objetivos

El objetivo del trabajo es el desarrollo de la cuarta versión de la aplicación HOMES.

La evolución consiste, por tanto, en la migración de la aplicación a dispositivos Android superiores a partir de Ice Cream Sandwich, como en la implementación de nuevas funcionalidades dentro del sistema.

La migración del sistema supone los siguientes retos:

- El desarrollo de una interfaz de usuario clara y estable mediante la API de Xamarin, adaptando los conocimientos de desarrollo en Android adquiridos durante la carrera.
- La compresión del sistema HOMES, dicho sistema se compone de la aplicación previa para Windows Mobile (desarrollado mediante el framework Compact .NET 2.0), y del back office que la respalda.
- La comunicación entre dicho back office, desarrollado mediante el framework .NET con la nueva aplicación Android.

Adicionalmente, la implementación de las nuevas funcionalidades requiere el análisis adecuado para su correcta integración en el sistema una vez éste sea migrado, además de la utilización de diferentes herramientas desconocidas hasta el momento para su desarrollo.

Las características del sistema, así como los datos que maneja, suponen la criticidad de aspectos como la robustez y la seguridad.

Todas estas condiciones implican que la realización de este proyecto va a poner a prueba y a desarrollar las aptitudes, tanto implícitas como explícitas, adquiridas durante la carrera.

1.3. Estructura del documento

Este documento refleja todas las fases por las que ha atravesado el proyecto. Se ha dedicado un capítulo para describir todos sus puntos clave y características.

- Capítulo 1 (***Introducción***): Se introducen los objetivos y alcance del proyecto.
- Capítulo 2 (***Estado del arte***): Se realiza un estudio acerca de las plataformas involucradas en el proyecto. Además se introduce el concepto de plataformas cruzadas.
- Capítulo 3 (***Análisis***): Se documenta el análisis realizado sobre HMCH, es decir, sobre la aplicación original y sobre su back office. Adicionalmente se formaliza la lista de requisitos del proyecto.
- Capítulo 4 (***Diseño de la aplicación***): Se muestra el diseño de la aplicación y de la base de datos.
- Capítulo 5 (***Desarrollo***): Se explicará el desarrollo de la aplicación poniendo énfasis al uso de Xamarin y otras herramientas auxiliares.

- Capítulo 6 (***Pruebas***): Se mostrarán las pruebas realizadas sobre la aplicación, diferenciando el ámbito de las mismas.
- Por último, el capítulo 7 (***Conclusiones y trabajo futuro***) incluirá las conclusiones obtenidas de la realización completa de la aplicación y se comentarán posibles trabajos futuros que pudieran aportar mejoras al proyecto software.
- ***Anexos****:* información para una mejor comprensión del documento que, por su extensión, se ha decidido no incluir como contenido de la memoria.

2. ESTADO DEL ARTE

En este capítulo se pretenden mostrar las características y estado actual de las diferentes tecnologías utilizadas en el proyecto, de manera que pueda esclarecer el motivo de la migración, además de aportar conocimiento sobre el uso de las plataformas cruzadas.

2.1. Homes

La aplicación HMS se desplegó para España a finales del año 2006 en una primera versión.

El siguiente año se desplegó en Portugal, correspondiendo con su segunda versión.

Finalmente en 2008 se desplegó la tercera versión, que se mantiene actualmente, ampliando el catálogo de países que utilizan este servicio, con la inclusión de Bélgica, Italia y Holanda.

Cabe destacar los años que lleva desplegada ésta última versión, siete años hasta el despliegue previsto para Homes 4.

2.2. Sistemas operativos móviles

En este apartado se pretende ilustrar el estado actual de los sistemas operativos implicados en el proyecto y tratar de concluir, mediante el análisis de sus características, el porqué de la elección de migración tomada por AL.

2.2.1. Android

En los últimos años Android se ha posicionado como el líder indiscutible en la cuota de mercado de Smartphones, seguido muy de lejos por iOS.

A pesar de que los últimos estudios de mercado muestran una ligera caída en la cuota de Android, ésta sigue siendo muy superior a la de sus competidores [(3)].

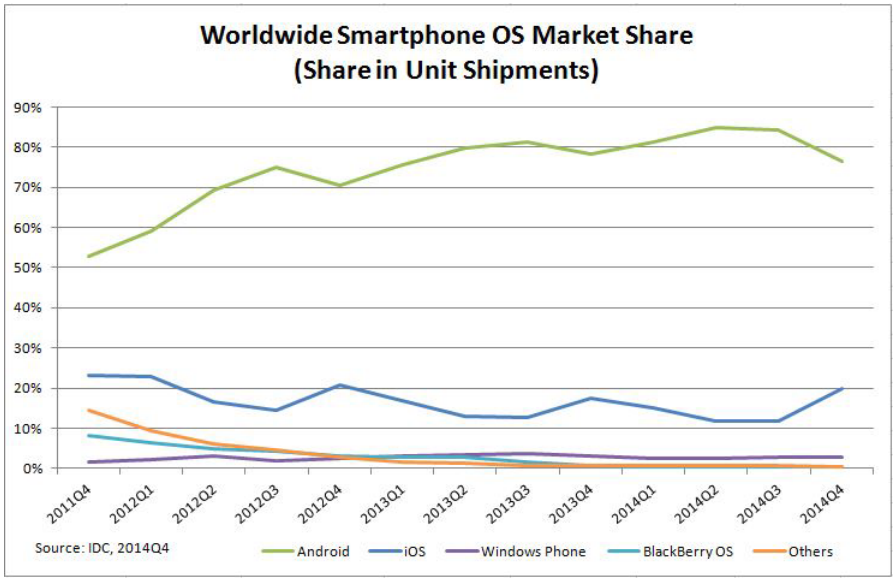

Figura 1: Cuota de mercado según S.O.

Si observamos las diferentes versiones disponibles de Android, resulta esclarecedor ver que la distribución del uso de versiones previas a Ice Cream Sandwich no llega al 10% del total.

De este modo es fácil comprender el porqué del uso de la versión 4.0 como versión objetivo [(4)].

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%

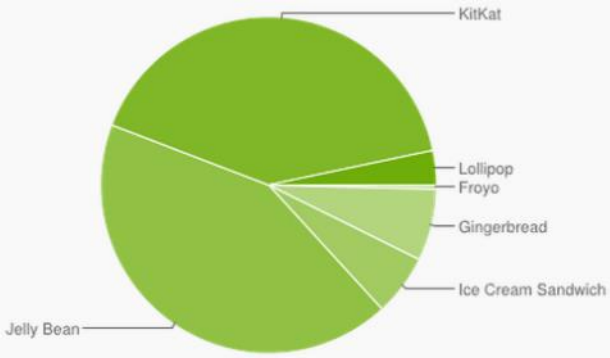

Figura 2: Distribución de versiones Android

Esto supone un gran aliciente a la hora de escoger Android como el objetivo de la migración, dado el uso tan extendido que posee este sistema operativo, que podrá facilitar el manejo de la aplicación por parte de los técnicos.

Gracias a esta cuota de mercado y a la compatibilidad del sistema operativo con hardware menos potente, Android también lidera en la diversidad de sus terminales, abarcando un abanico de precios muy amplio.

2.2.2. Windows Mobile/Phone

En noviembre de 1996 Windows lanzó su plataforma para sistemas compactos (Windows CE); este sistema operativo requería muy poca memoria para ejecutarse, de manera que era perfecto para utilizarse en terminales móviles.

Esta primera versión era demasiado tosca y no lograba su objetivo de lograr un dispositivo portátil basado en pantalla táctil resistiva.

En Abril del año 2000 aparece Pocket PC 2000, basándose en la versión 3.0 de la plataforma CE; sería la primera versión de los sistemas denominados Windows Mobile.

Estos sistemas irían evolucionando junto con la plataforma CE, obteniendo un gran protagonismo en los fabricantes de terminales industriales.

Finalmente, Microsoft decidió descontinuar Windows Mobile a partir de la versión 6.0 (basada en Windows CE 5.0) para centrarse en Windows Phone. Con este nuevo sistema Microsoft pretendía popularizar un sistema multiplataforma, enfocándose en el mercado de consumo.

Windows Phone no sólo términó siendo un fracaso frente a sus competidores (iOS y especialmente Android), sino que además no permitía la retrocompatibilidad con los antiguos Windows Mobile, al basarse en un nuevo paradigma denominado Silverlight [(5)].

Los fabricantes de terminales industriales encontraron esta situación insostenible, dado que los sistemas operativos Windows Mobile no tenían soporte, y además la migración a sistemas Windows Phone no era atractiva debido a su coste y a que se trataba de un entorno poco atractivo. Esto forzó a Microsoft a lanzar Windows Mobile 6.5.

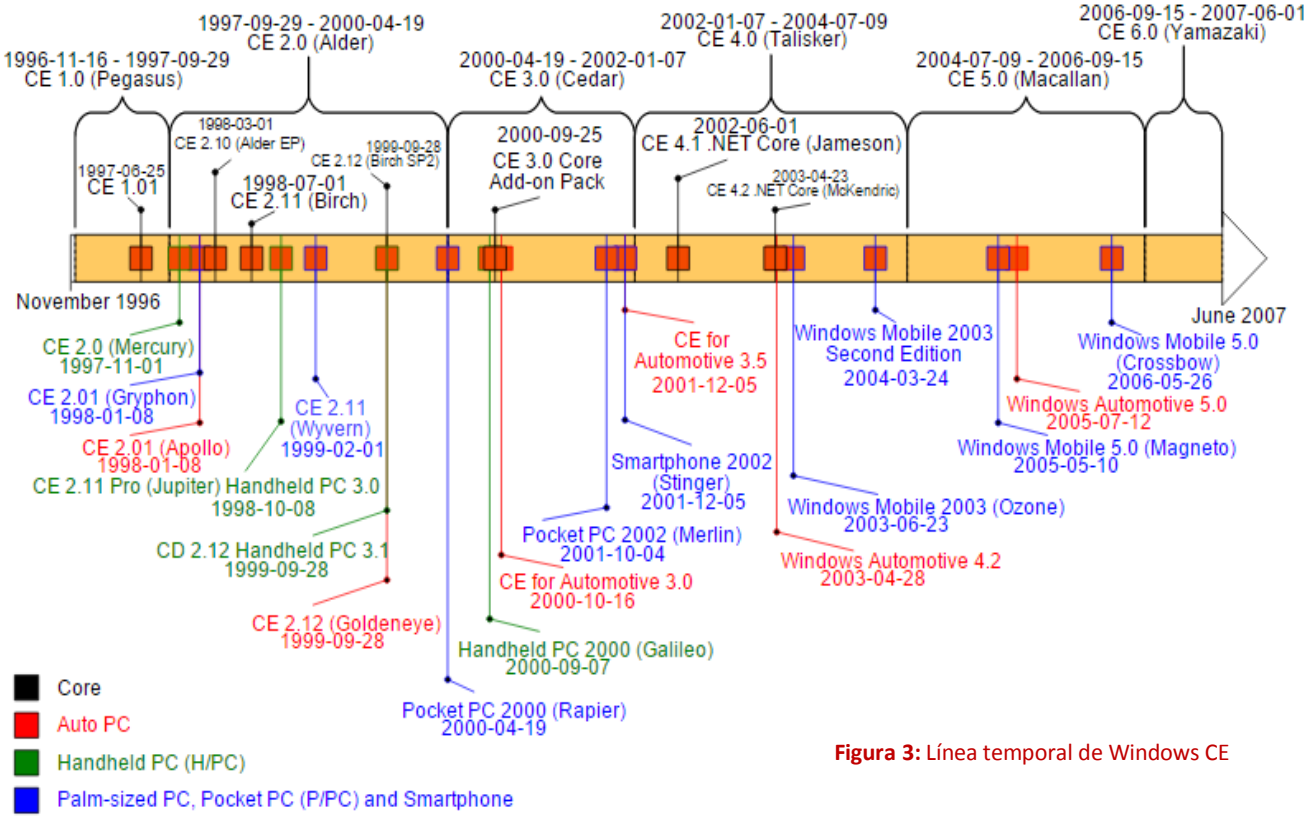

Figura 3: Línea temporal de Windows CE

El lanzamiento de Windows Mobile se puede ver desde los ojos de la línea temporal de Windows Mobile como una rama alternativa de la misma, entendida como una transición entre Windows Mobile y Windows Phone, algo que a día de hoy no se ha producido.

Tras analizar el estado actual de los sistemas de Microsoft, la migración a Windows Phone no resulta tentadora, debido al esfuerzo que requiere y a la desconfianza que proporciona dicho sistema.

2.3. Xamarin

Xamarin es una plataforma que permite el desarrollo de aplicaciones multiplataforma mediante el lenguaje de programación C#.

Xamarin surge del proyecto Mono; este proyecto permite la ejecución de una serie de herramientas compatibles con .NET en sistemas operativos GNU/Linux. Tras la adquisición de Mono por Novell, algunos empleados de Mono fundaron Xamarin [(6)].

Xamarin permite escribir aplicaciones para iOS, Android o Windows Phone mediante C#; esto encaja a la perfección con las necesidades de la empresa, ya que al soportar .NET todo el back office de la aplicación es compatible y, además, los trabajadores tienen experiencia y formación en este lenguaje.

Además, al permitir también el desarrollo de aplicaciones iOS y Windows Phone, las posibles migraciones supondrían una reducción de trabajo.

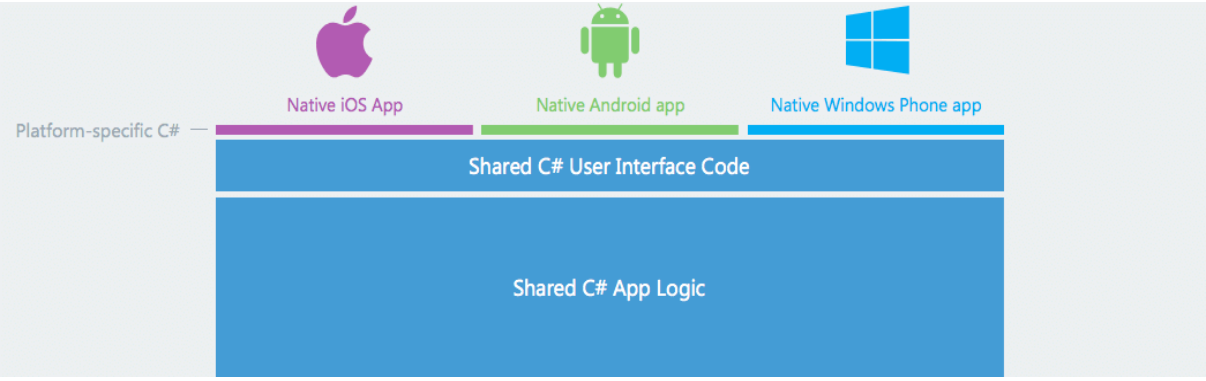

Figura 4: Plataformas Xamarin

Respecto a la fidelidad de Xamarin frente a los entornos nativos que soporta, en su página oficial aseguran proveer el 100% de sus APIs y un rendimiento similar a las aplicaciones desarrolladas nativamente [(7)].

En el caso de Android existen una gran cantidad de librerías Java de terceros; para poder utilizar estas librerías Xamarin permite el "atado" (Binding) de las mismas.

3. ANÁLISIS

En esta sección se pretende documentar el análisis previo al diseño de la aplicación.

Durante dicho análisis intentaremos comprender la aplicación inicial, estudiar el back office heredado que se utilizará en la nueva versión y formalizar el catálogo de requisitos.

3.1. Análisis de la aplicación previa

A la hora de realizar el análisis de una evolución tecnológica es vital realizar antes el análisis de las versiones previas del sistema pendiente de evolución.

Este análisis cumple el objetivo de recolectar información para formalizar los requisitos de la nueva aplicación.

La versión previa de HOMES está formada por los siguientes apartados:

* Resumen de DNs: Muestra un resumen de todas las DNs que el técnico debe resolver. Toda la información mostrada procede del servidor central. Presenta tanto las visitas del técnico como su stock.
* Empezar/finalizar ruta: Gestiona el comienzo de una ruta (se almacena fecha y hora de inicio), y el final de una ruta (se almacena fecha y hora de finalización y se envían los datos al servidor).
* Visitas: Permite la gestión de todas las visitas definidas en la ruta del técnico (lista de DNs).
* Sincronización: Comprende la comunicación de datos (subida y bajada) con el servidor.

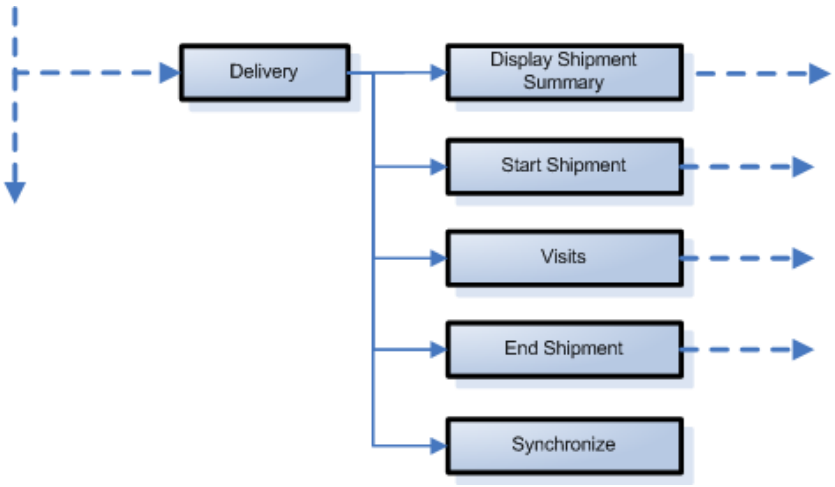

Figura 5: Esquema HMS 3

Tabla 3: TIPOS DE PRODUCTO				
PRODUCTO	CILINDRO	DISPOSITIVO	CONSUMIBLE	SERVICIO
-Código de barras	Si	Si	No	No
- Número de lote	Si	No	Si	No

El estudio de las funcionalidades de la aplicación previa puede observarse detalladamente en el Anexo C.

3.2. Back Office

Para realizar una nueva versión de una aplicación existente resulta crucial comprender y saber manejar el back office que la respalda. En este apartado se pretenden mostrar, de forma global, los componentes que componen dicho back office y cómo se interrelacionan entre ellos.

Lo primero que conviene señalar es que el Back Office encargado de respaldar la versión previa de la aplicación está desarrollado en C# mediante el framework .NET, que se encarga de comunicarse con la base de datos mediante instrucciones SQLServerMobile.

Para el respaldo de la aplicación el back office está formado por los siguientes componentes:

- Capa de negocio.
- Capa de acceso a datos.
- Capa de datos.
- Capa de manejo HTTP.

Cada una de estas capas serán definidas con mayor precisión en el anexo D: Análisis del Back Office incluido al final de este trabajo.

3.3. Análisis de requisitos

Mediante el análisis de requisitos se pretenden definir las funcionalidades que debe alcanzar el proyecto. A lo largo de este análisis se acotarán el comportamiento y restricciones del sistema.

Los requisitos han sido formalizados mediante el estudio de la aplicación objeto de migración y mediante los documentos del cliente acerca de las nuevas funcionalidades.

3.3.1. Requisitos funcionales

Los requisitos funcionales del sistema muestran las funcionalidades que éste debe cumplir.

3.3.1.1 Requisitos funcionales generales

- **RF1**: Adaptar las funcionalidades de la versión previa a Android (véase Anexo C: *Análisis de la versión previa*).
  - RF1.1: *Sincronización*: La sincronización se podrá hacer en cualquier momento de la ruta (véase RF5).
  - RF1.2: *Resumen de DNs*: Este apartado se elimina en la nueva versión. El stock del técnico se mostrará en un apartado distinto.
  - RF1.3: *Inicio y finalización de rutas*: Este apartado se elimina en la nueva versión.
  - RF1.4: *Visitas*: El módulo de visitas debe mantener la funcionalidad de la versión previa, mostrando además la información del módulo Resumen (véase 3.1.2):
    - RF1.4.1: Se mostrará la información de DNs del módulo resumen en la lista de detalles de visita (véase 3.1.4.1).
    - RF1.4.2: Se mostrará la información de líneas de detalle del módulo resumen en la lista de detalles de servicio, manteniendo todas sus funcionalidades (véase 3.1.4.2).
- **RF2**: Encriptar todas las comunicaciones mediante HTTPS.
- **RF3**: Habilitar conexión 3G/GPRS con el punto de acceso dedicado de AL.
- **RF4**: Habilitar conexión Wifi en las plantas AL/VA.
- **RF5**: Habilitar sincronización bajo demanda; bajo esta nueva modalidad el sistema debe ser capaz de gestionar la descarga de cualquier actualización de DN, además de la subida de las mismas, sin necesidad de encontrarse en una planta VA.
- **RF6**: El sistema debe permitir el uso de software GPS de terceras partes; esta funcionalidad solo estará disponible para los clientes que compren las licencias de dicho software. Actualmente se implementará el software de GPS Sygic.
- **RF7**: Los ajustes de la aplicación permitirán la selección entre dos direcciones predefinidas (ambas editables) para producción y preproducción. Es necesario que al cambiar dicha selección se eliminen automáticamente todos los datos de la aplicación almacenados en el terminal.
- **RF8**: Aunque la aplicación permite múltiples idiomas para el usuario, también debe permitir el manejo de dos idiomas para un mismo país. Pese a que la mayoría de países tienen un único idioma definido en el servidor central y que los ajustes de la aplicación permiten cambiarlo, en países como Bélgica es necesario mostrar determinada información en el idioma del paciente, que puede ser distinto al del operario.
  - RF8.1: La aplicación diferenciará entre el idioma del usuario y del paciente.

- RF8.2: El usuario debe ser capaz de cambiar el idioma de la aplicación sin necesidad de ninguna sincronización (y por lo tanto sin necesidad de conexión) mediante los diccionarios disponibles.
- RF8.3: La aplicación debe gestionar la descarga de los referenciales dinámicos de la aplicación en diferentes idiomas y almacenarlos localmente, de modo que tras la sincronización se puedan mostrar los referenciales para cualquier idioma que el usuario seleccione sin necesidad de conexión.
- RF8.4: Se debe mostrar el contenido dinámico de la aplicación (la información descargada del servidor dependiente de cada usuario) en el idioma del usuario.
- RF8.5: La aplicación debe mostrar el contenido dinámico de la eDN, o de los cuestionarios terapéuticos en el idioma del paciente.

➢ **RF9**: Tras doce horas de inactividad, la aplicación debe acceder directamente a la pantalla de login por motivos de seguridad.

➢ **RF10**: Búsqueda filtrada: En todas las listas de la aplicación debe permitirse el filtrado en tiempo real por descripción, o en los casos que exista por descripción e id del producto.

➢ **RF11**: Lectura mediante escáner lineal: Se debe permitir la lectura de códigos de barra donde sea necesario mediante el lector lineal, en caso de utilizarse el terminal objetivo (Motorola TC55), para todas las entradas donde se espere un código de barras (número de lote, código de barras, etc.).

- RF11.1: Se debe permitir la lectura tanto mediante un botón software, como con el botón hardware del dispositivo.
  - RF11.1.1: El botón software estará asociado a un campo de texto, donde se mostrará el resultado de la lectura.
  - RF11.1.2: El botón hardware sólo activará el láser cuando se pulse con el cursor (foco) en un campo de texto que permita la lectura laser.
- RF11.2: Si el terminal no dispone del hardware necesario, se debe deshabilitar esta funcionalidad.
- RF11.3: El escáner no debe estar operativo, salvo para leer los campos que lo requieran.
- RF11.4: El escáner se desactivará si se pulsa el botón físico o software mientras está activado, al salir de la pantalla que lo requiere, tras leer correctamente o tras un tiempo de espera predefinido.

3.3.1.2 Requisitos funcionales de DN

Se definen las funcionalidades del sistema necesarias para las notas de entrega y sus líneas asociadas (véase Anexo C: *Análisis de la versión previa*).

➢ **RF12**: El usuario podrá completar diversos formularios terapéuticos con la colaboración del paciente dentro de una DN.

- **RF13**: El usuario podrá completar diversos formularios técnicos respecto a determinados productos dentro de una DN.
- **RF14**: El usuario podrá descargar una DN de otro técnico, indicando su id, si dicha DN no está finalizada y añadirla a su lista de DNs.
- **RF15**: El usuario podrá buscar una DN por su id dentro de la lista de DNs y acceder a ella directamente
- **RF16**: El usuario podrá filtrar la lista de DNs según su estado (todas, finalizadas y sin finalizar).
- **RF17**: El usuario podrá ordenar la lista de DNs según diferentes parámetros como la fecha o el nombre del paciente.
- **RF18**: El usuario podrá indicar una transferencia de líquido entre dos productos de tipo cilindro (véase 3.1, tabla 3) dentro de una DN.
- **RF19**: El usuario podrá añadir o eliminar productos al stock del cliente de una DN.
- **RF20**: Capturar las coordenadas GPS de la residencia del paciente dentro de una DN.
- **RF21**: eDN. El sistema debe poder gestionar el proceso de generación y representación de notas de entrega electrónicas (véase anexo A).
  - RF21.1: La eDN debe mostrar toda la información referente a la visita y al paciente, es decir debe mostrar la misma información que tuviera la nota en papel.
  - RF21.2: La aplicación debe poder capturar las firmas, tanto del usuario (técnico de AL), como del signatario (paciente o representante del paciente). Del mismo modo debe reflejarse en la eDN la imposibilidad o negación de firma por parte del signatario.
  - RF21.3: Se permitirá mostrar una vista previa de la eDN, antes de capturar la firma y nombre del signatario, y tras completar todos los campos.
  - RF21.4: Se firmará digitalmente el documento final con un certificado de AL antes de cerrar la DN.
- **RF22**: Toma de fotografías. Para una DN, el sistema debe gestionar la captura y subida de fotos de seguridad. Esto permitirá al usuario de HMS Android tomar fotos de incidencias en sus visitas, así como de capturar la nota en papel firmada por el paciente.
- **RF23**: La aplicación debe permitir al usuario añadir notas a la DN, estas notas subirán al servidor con su DN.
- **RF24**: La aplicación debe permitir al usuario cerrar una DN ante la imposibilidad de completarla, indicando el motivo de dicha imposibilidad.
- **RF25**: El usuario podrá comprobar la posología del paciente de una determinada DN.

- **RF26**: Multilíneas. El sistema debe poder manejar lecturas de varios códigos de barra de cilindro para una misma línea de una DN. Esta funcionalidad es necesaria debido a la situación de Italia y Bélgica de entregar o recoger gran cantidad de cilindros en una misma visita.
  - RF26.1: El sistema debe crear una sublínea por cada cilindro capturado.
  - RF26.2: La línea maestra se debe deshabilitar, no permitiendo su edición.
  - RF26.3: Las sublíneas deben subir al servidor como líneas individuales.
  - RF26.4: El sistema debe seguir permitiendo las líneas individuales en los casos que sea necesario.
  - RF26.5: La existencia de multilíneas debe ser configurable por país cliente, apareciendo por defecto para Italia y Bélgica, pudiendo aplicarse a más países en el futuro si lo requieren y cumplen las condiciones.
- **RF27**: Campos extra en líneas de detalle de la DN: Se deben añadir los siguientes campos a la tabla de la línea de detalle en la base de datos, y aplicar las siguientes reglas a cada uno de ellos:
  - RF27.1: Número de lote pre-asignado: Si un número de lote pre-asignado está presente para una línea de detalle, éste debe asignarse al número de lote definitivo, a la espera de que el usuario lo valide o lo sobrescriba con otro valor.
  - RF27.2: Código de barras pre-asignado: Si un código de barras pre-asignado está presente para una línea de detalle, éste debe asignarse al código de barras definitivo, a la espera de que el usuario lo valide o lo sobrescriba con otro valor.
  - RF27.3: Fecha límite pre-asignada: Si una fecha límite pre-asignada está presente para una línea de detalle, debe mostrarse en dicha línea y en la eDN.
- **RF28**: Códigos AIC: Por motivos legales, las descripciones de algunos productos relacionados con oxígeno deben ser sustituidas para determinados países cliente, por su correspondiente código AIC, estos códigos deben gestionarse según las siguientes reglas:
  - RF28.1: Debe añadirse una tabla con la información de los códigos AIC (descargados al sincronizar, véase RF1.1), y una columna con el código AIC asociado a cada línea en la tabla de líneas de detalle.
  - RF28.2: Debe existir un parámetro en la tabla de configuración que permita activar o desactivar esta funcionalidad.
  - RF28.3: Siempre que se muestre un producto relacionado con oxígeno en la aplicación debe sustituirse su descripción con la descripción asociada a su código AIC si es una recogida de producto, o por la descripción asociada al código F si es una entrega.
  - RF28.4: Si no se encuentra descripción asociada para un producto en la tabla de códigos AIC, se mostrará la descripción habitual.

- **RF29**: Añadir un dispositivo inesperado a la lista de líneas de detalle de una DN:
  - RF29.1: Al introducir el código de barras en una línea de detalle inesperada, asociada a un dispositivo (véase 3.1.4.5) se debe llamar al WebService correspondiente (implementado por terceros) y recuperar el id del producto recibido.
  - RF29.2: Si el WebService no devuelve ningún Id, se devuelve el id GEN, asociado a un producto desconocido.
- **RF30**: Autofill: El usuario podrá completar automáticamente las líneas de detalle que no requieran la captura de código de barras.
  - RF30.1: Autofill general: Desde la lista general de líneas de detalle el autofill debe iterar sobre todas ellas validando las que sean posibles.
  - RF30.2: Autofill a nivel de línea: Desde cada línea el autofill debe completar los valores necesarios y delegar la responsabilidad de validar en el usuario.
  - RF30.3: Al añadir cualquier línea que esté fuera de stock del camión (véase Anexo C, Apartado C.2, stock de técnico) se debe bloquear el proceso a la espera de confirmación del usuario.
  - RF30.4: Sólo se podrá realizar autofill de productos que no requieran la lectura del código de barras (véase 3.1, tabla 3).
- **RF31**: Diferenciar gráficamente las entregas de las recogidas de producto en la lista de líneas.
- **RF32**: El usuario podrá cerrar una DN y enviarla al servidor, junto con su correspondiente eDN firmada (véase RF21).

3.3.2. Requisitos no funcionales

- **RNF1**: Se deben proteger la confidencialidad de los datos de los pacientes.
- **RNF2**: La aplicación debe poder ser funcional en cualquier dispositivo Android cuya versión sea igual o superior a Ice Cream Sandwich.
- **RNF3**: La interfaz de usuario debe ser clara y usable. Es importante mostrar la mayor cantidad de información posible para minimizar la necesidad del usuario de consultar diferentes pantallas.
- **RNF4**: El tiempo de respuesta en las llamadas al servidor y a la base de datos local debe ser razonable (nunca superior a un minuto).
- **RNF5**: La eDN (véase RF21) debe estar firmada con un certificado válido para demostrar la integridad del documento.

4. DISEÑO DE LA APLICACIÓN

4.1. Componentes del proyecto. Análisis de tecnologías

El sistema está formado por los siguientes componentes:

- Aplicación cliente Android.
- Servidor HOMES.
- Base de datos interna.

La aplicación accede tanto al servidor HOMES como a la base de datos interna como muestra el siguiente esquema:

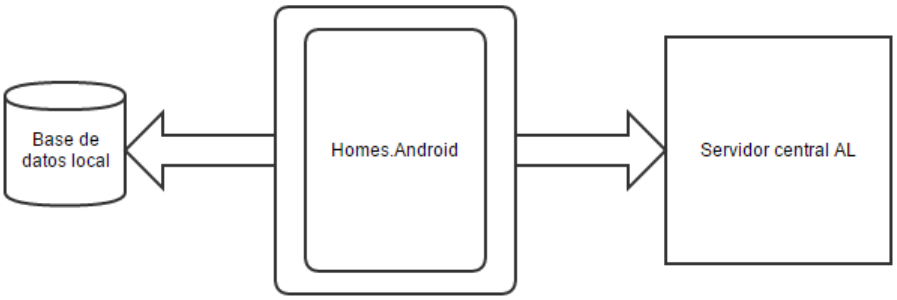

Figura 6: Componentes del sistema

4.1.1. Aplicación cliente Android

El objetivo de la aplicación es el seguimiento de las visitas que conforman la ruta asociada a un técnico.

El técnico se identificará en la aplicación y podrá acceder a la ruta que tenga asociada en el servidor.

A continuación se expone el diagrama de flujo de la aplicación:

4.1.1.1. Login

Al arrancar la aplicación se muestra la pantalla de login. Si los datos no son correctos, o hay algún fallo a la hora de realizar la conexión, aparece un mensaje de error. Desde esta pantalla es posible acceder a los ajustes de la aplicación (véase 4.1.1.2).

El usuario podrá hacer login online, o bien si ya lo hizo previamente y de este modo descargó la información necesaria del servidor, podrá hacer login offline.

Si el login es correcto aparecerá la pantalla principal (véase 4.1.1.3).

Figura 7: Login

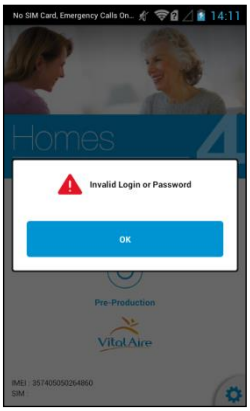

Figura 8: Login incorrecto

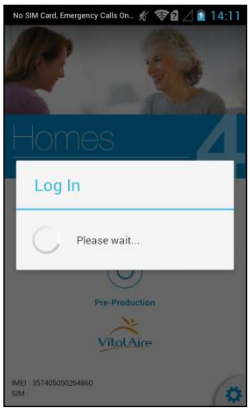

Figura 9: Entrando

Durante el proceso de login, el sistema descargará del servidor los datos referenciales (datos estáticos sobre productos, configuración de usuario…), y los almacenará en la base de datos local.

4.1.1.2. Ajustes

El usuario podrá acceder a los ajustes de la aplicación desde la pantalla de login y desde el menú principal (véase 4.1.1.3).

Desde la pantalla de ajustes, el usuario podrá llevar a cabo las siguientes acciones:

4.1.1.2.1. Culturas

Seleccionar el idioma de usuario de la aplicación, o bien usar el idioma predefinido por configuración. Una vez aplicados los cambios se reiniciará la aplicación.

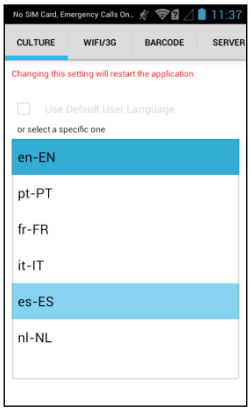

Figura 10: Idioma

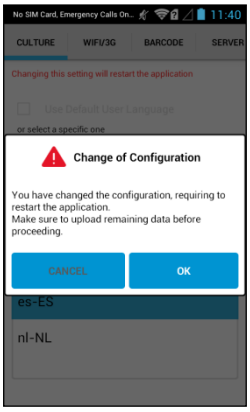

Figura 11: Aceptar cambio

Figura 12: HMS Español

Nótese que no existe la posibilidad de mostrar el panel lateral, ni de seleccionar el idioma por defecto al acceder a los ajustes desde la pantalla de login. Esto último se debe a que el idioma por defecto se encuentra almacenado en la base de datos interna para cada usuario, obteniendo su valor al descargar los referenciales en el proceso de login (véase 4.1.1.1).

Esto cambia al acceder a los ajustes desde dentro de la aplicación, tras identificarse. Al realizar los cambios siempre se reiniciará la aplicación.

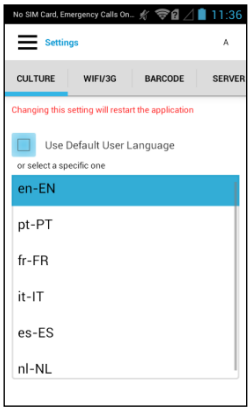

Figura 13: Idioma predefinido

Figura 14: Reinicio de HMS

4.1.1.2.2. Conectividad

Activar/desactivar la conexión Wifi y 3G.

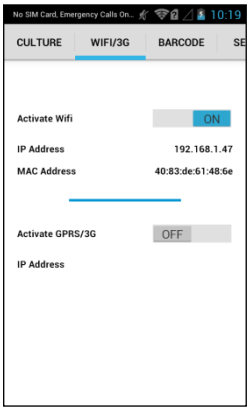

Figura 15: Wifi/3G

4.1.1.2.3. Prueba de lector

Testar el lector lineal laser de códigos de barra de su terminal (sólo activado para el terminal objetivo, Motorola TC55).

Figura 16: Prueba Lector

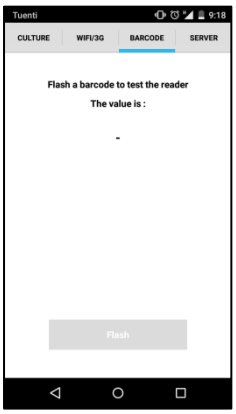

Figura 17: Sin Lector

4.1.1.2.4. Servidor

Seleccionar la dirección del servidor. El usuario podrá seleccionar entre la dirección de producción o pre-producción, o bien editar manualmente la dirección. Una vez aplicados los cambios, la aplicación borrará los datos locales (por petición explicita del cliente), y se reiniciará la aplicación. Para acceder a esta pantalla es necesario introducir un código de acceso.

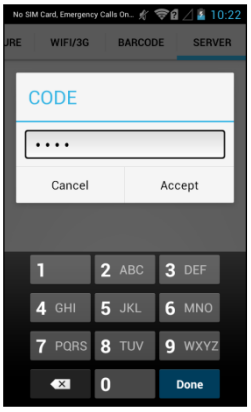

Figura 18: Código de acceso

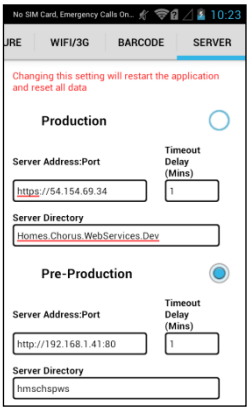

Figura 19: Ajuste de servidor

4.1.1.3. Menú principal

Tras identificarse correctamente, el usuario tendrá acceso al menú principal, desde el cual, mediante un panel desplegable, podrá navegar entre los diferentes contenidos de la aplicación.

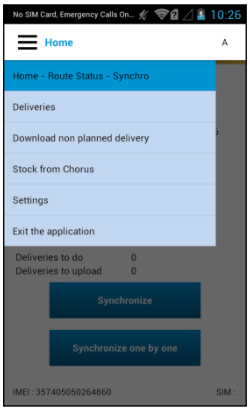

Figura 20: Menú lateral

4.1.1.4. Sincronización

La aplicación abrirá la ventana de sincronización por defecto cuando el usuario se identifique correctamente. Desde esta ventana el usuario podrá sincronizar sus DNs con el servidor central. El proceso de sincronización comenzará subiendo al servidor todas las DNs completadas y finalizará bajando las DNs asociadas a la ruta del usuario.

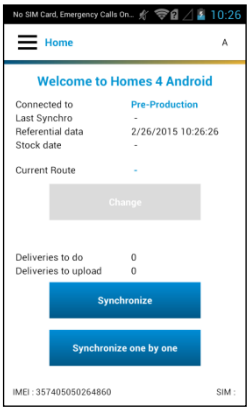

Figura 21: Sincronización

Figura 22: Sincronizando

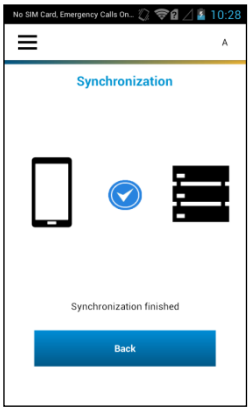

Figura 23: Sincronizado

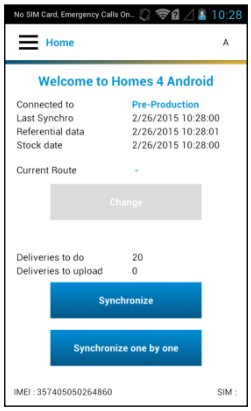

Figura 24: Resumen de Sincronización

Existe la posibilidad tanto de sincronizar todas las DNs a la vez o de sincronizarlas una a una.

Figura 25: Sincronización global

4.1.1.5. Descarga de DN inesperada

Pese a que el técnico tenga asociada una ruta con una serie de DN por realizar, se ha añadido la posibilidad de añadir una DN, de este modo los usuarios podrán intercambiar DNs si la situación lo exige.

Para descargar una DN inesperada, el usuario deberá introducir el número de la misma. La aplicación informará del éxito de la operación, y en caso favorable, añadirá la DN a la lista asociada al usuario.

Figura 26: Descarga DN

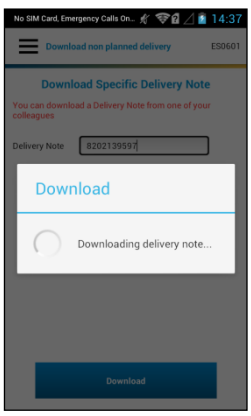

Figura 27: Descargando DN

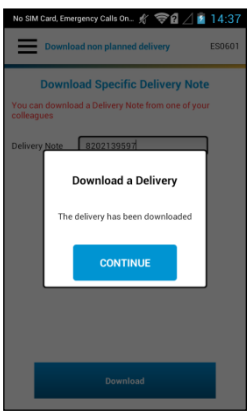

Figura 28: Descarga completa

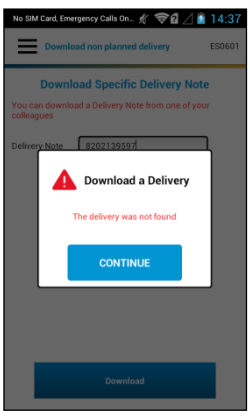

Figura 29: Descarga fallida

Tras añadirse la DN inesperada, se mostrará en la lista con una etiqueta que indique su novedad, dicha etiqueta desaparecerá cuando el usuario haya entrado una vez en dicha DN.

Figura 30: Nueva DN

Figura 31: DN Visitada

4.1.1.6. Stock de camión

En este apartado el usuario podrá comprobar el stock oficial del que dispone. Esta información es estática y no puede editarse.

La información de stock se descarga al sincronizar (véase 4.1.1.4), la información está ligada a la ruta del técnico y puede aparecer vacía o desactualizada si éste no se sincronizó.

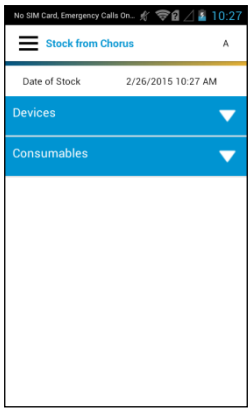

Figura 32: Stock de camión [1]

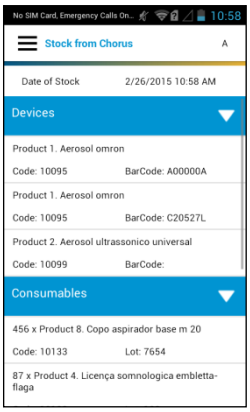

Figura 33: Stock de camión [2]

4.1.1.7. Deliveries

Este es el apartado vital de la aplicación, desde el mismo el usuario podrá gestionar las visitas asociadas a su ruta.

La lista de DNs se descarga al sincronizar (véase 4.1.1.4), esta información está ligada a la ruta del técnico y puede aparecer vacía o desactualizada si este no sincronizó cuando debía.

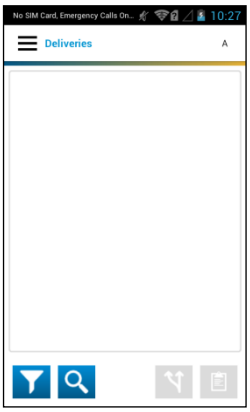

Figura 34: Lista sin DNs.

Figura 35: Lista con DNs.

4.1.1.7.1. Notas del paciente

Al pulsar el icono de cada DN, el usuario podrá ver las notas asociadas al paciente de esa DN, dichas notas bajan del servidor central y no pueden ser editadas.

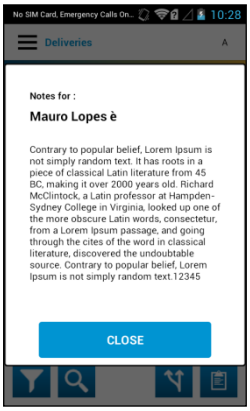

Figura 36: Notas del paciente

4.1.1.7.2. Ordenar y filtrar

En esta sección el usuario podrá ordenar las DN según los criterios disponibles, y filtrarlas dependiendo de su estado de completación-

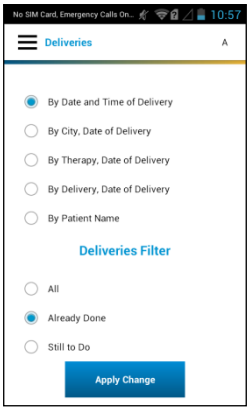

Figura 37: Ordenar y filtrar

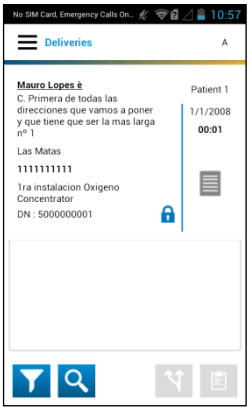

Figura 38: DNs completas

4.1.1.7.3. Búsqueda de DN

El usuario podrá buscar una DN por su número, si se encuentra se abrirá directamente dicha DN (véase 4.1.1.8).

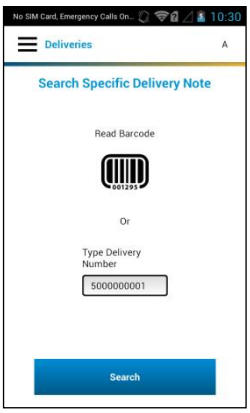

Figura 39: Búsqueda de DN

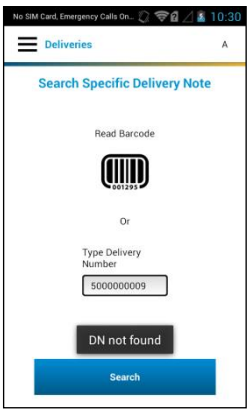

Figura 40: DN inexistente

Figura 41: DN encontrada

4.1.1.7.4. Navegación

Esta opción permite al usuario desplegar el navegador GPS Sygic (la aplicación mostrará las opciones disponibles si este navegador no está instalado en el dispositivo) mostrando la ubicación del usuario.

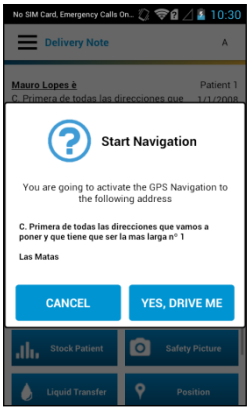

Figura 42: Comenzar navegación GPS.

4.1.1.7.5. Abrir DN

El usuario accede a la DN seleccionada (véase 4.1.1.8).

4.1.1.8. DN

Desde este apartado el usuario podrá gestionar una DN en particular.

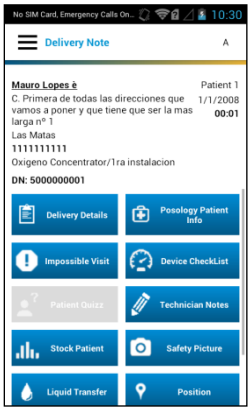

Figura 43: Menú DN [1]

Figura 44: Menú DN [2]

Podrá acceder a los siguientes apartados:

4.1.1.8.1. Detalles de entrega

En este apartado se muestran las líneas de detalle de una DN, permitiendo editarlas, autocompletarlas y editarlas individualmente.

Al editar una línea se mostrarán los elementos de la misma en detalle, y permitirá al usuario introducir la información necesaria dependiendo del tipo de producto de la línea (véase 3.1, tabla 3). Para navegar entre líneas el usuario podrá desplazar lateralmente la pantalla visitando las diferentes líneas de detalle de la DN, o bien validando las líneas una a una.

Al validar una línea, si los datos introducidos son correctos, se almacenarán, y la pantalla se desplazará a la siguiente línea de detalle.

El usuario podrá introducir los datos a mano, o bien mediante el lector de códigos de barra (pulsando el icono de código de barras).

Si lo introduce manualmente, se informará si el formato es incorrecto.

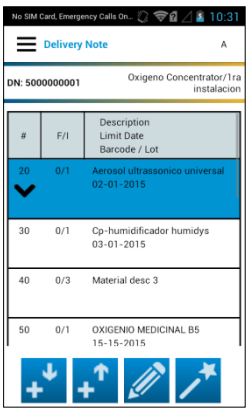

Figura 45: Líneas

Figura 46: Editar línea [1]

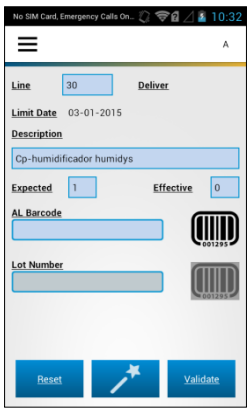

Figura 47: Editar línea [2]

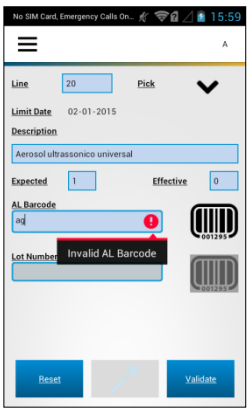

Figura 48: Editar línea [3]

En Bélgica e Italia, algunas visitas pueden tener más de cuarenta cilindros llenos, y otros tantos vacíos, de modo que se requiere poder capturar sus códigos de barra dentro de una misma línea, apareciendo éstos como sublíneas. En este caso el navegador de líneas lo mostrará como una única línea, pero al validar incrementará el número efectivo de la línea en vez de avanzar a la siguiente. Adicionalmente se mostrará una lista con las sublíneas capturadas.

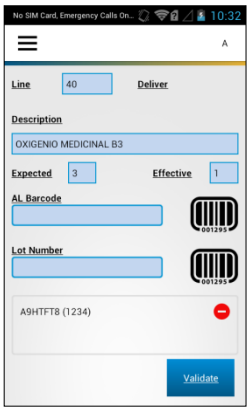

Figura 49: Editar multilínea [1]

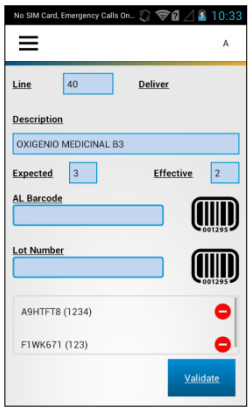

Figura 50: Editar multilínea [2]

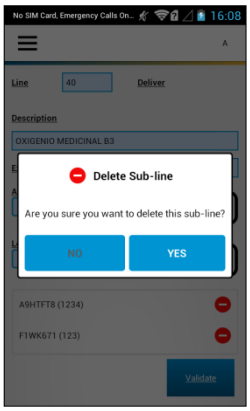

Figura 51: Eliminar sublínea

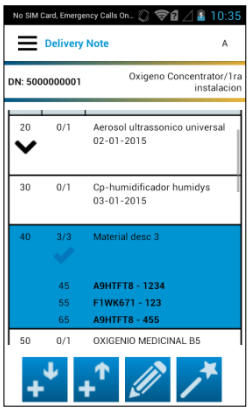

Figura 52: Resumen multilínea

El usuario podrá añadir líneas tanto de entrega como de recogida, además de especificar el tipo de producto que añadirá desde el menú de líneas.

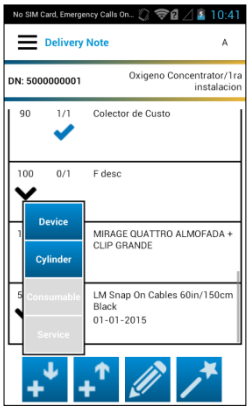

Figura 53: Añadir líneas

Al añadir una línea aparecerá una ventana similar al navegador de líneas visto al editar, sin embargo esta ventana no puede desplazarse lateralmente.

En este caso el usuario deberá añadir el producto de la línea, siendo optativo rellenar el código de barras y el número de lote, ya que se podrá editar más adelante.

A la hora de añadir el producto, el técnico tiene a su disposición un filtro con dos criterios de búsqueda. Mediante este filtro se podrá buscar el producto mediante el nombre del mismo, o mediante su identificador de negocio.

Además se permite realizar una búsqueda mixta, donde el usuario podrá introducir un indicio de ambos criterios, para separar un criterio del otro se utilizará el espacio. Ambos criterios pueden intercambiar su posición.

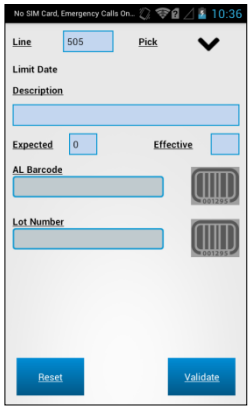

Figura 54: Añadir línea [1]

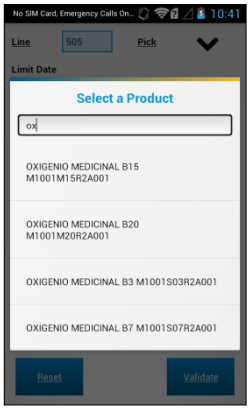

Figura 55: Añadir línea [2]

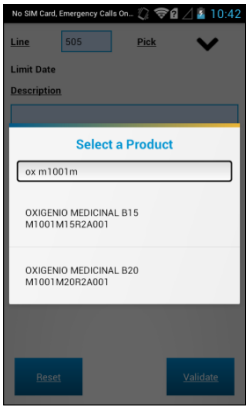

Figura 56: Añadir línea [3]

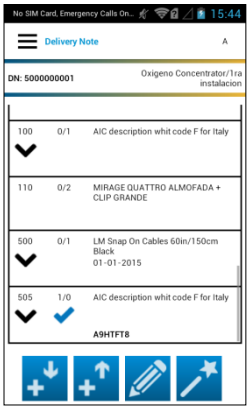

Figura 57: Añadir línea [4]

Al añadir dispositivos (véase 3.1, tabla 3), el usuario podrá añadir un dispositivo que no se encuentre en el stock, buscando el identificador del producto en el servidor central (véase 3.3.1.2, RF29)

El usuario podrá eliminar únicamente las líneas añadidas por él, nunca las líneas que descargó al sincronizar.

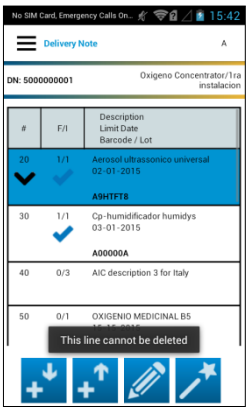

Figura 58: Eliminar línea [1]

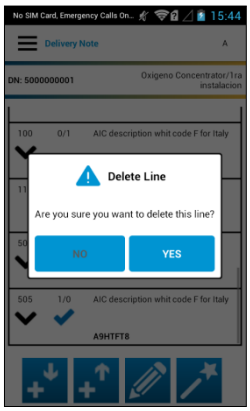

Figura 59: Eliminar línea [2]

Por último, el usuario podrá autocompletar las líneas que lo permitan (véase 3.3.1.2, RF30), al pulsar el icono de varita mágica, si se pulsa dentro de la lista de líneas, se iterará cada línea y se completarán las posibles, sin embargo si se pulsa dentro de la edición de cada línea solo se autocompletará dicha línea.

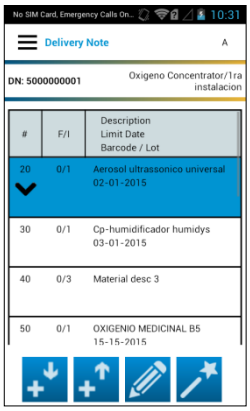

Figura 60: Autofill [1]

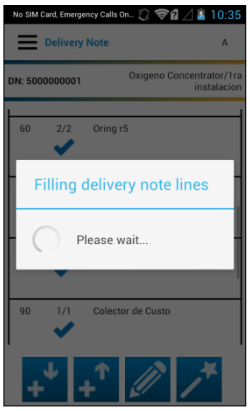

Figura 61: Autofill [2]

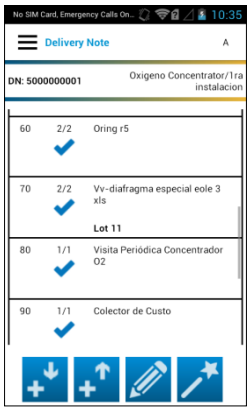

Figura 62: Autofill [3]

4.1.1.8.2. Posología

Muestra la posología e información del paciente asociado a la DN.

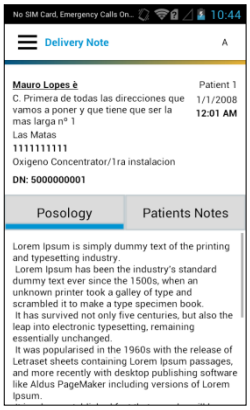

Figura 63: Posología

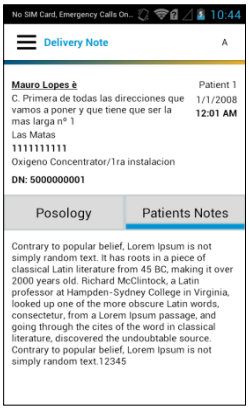

Figura 64: Notas

4.1.1.8.3. Visita imposible

Permite cerrar una visita debido a la imposibilidad de su resolución indicando la causa. Una visita con líneas completadas no se podrá cerrar por imposibilidad nunca.

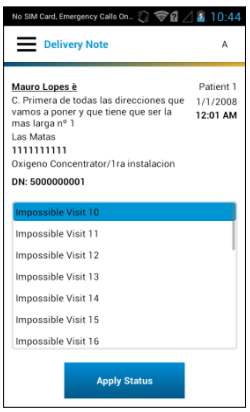

Figura 65: Visita imposible [1]

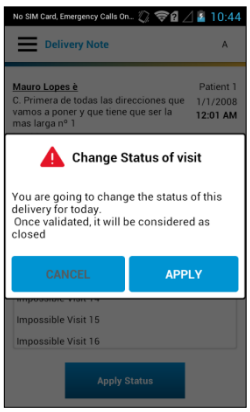

Figura 66: Visita imposible [2]

4.1.1.8.4. Cuestionario de dispositivo

El usuario podrá rellenar diferentes cuestionarios para un determinado dispositivo.

Una vez se indique el código de barras del dispositivo al que se le va a hacer el chequeo, se debe indicar el producto, para ello se contará con un filtro similar al descrito en detalles de entrega (véase 4.1.1.8.1), salvo que a diferencia de éste último, solo se tiene en cuenta un criterio, éste es el tipo de producto.

Se mostrarán los diferentes cuestionarios disponibles para el tipo de producto dado.

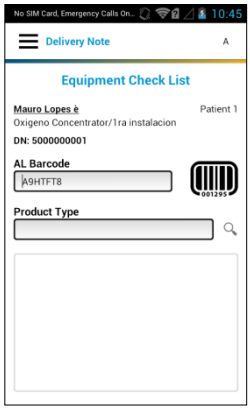

Figura 67: Cuestionarios de dispositivo [1]

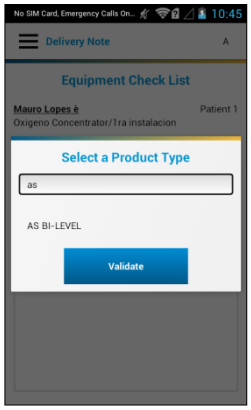

Figura 68: Selección de dispositivo

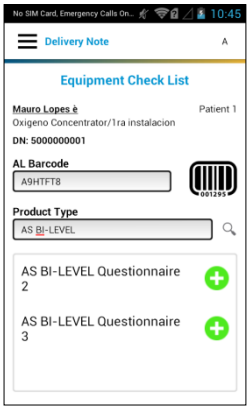

Figura 69: Cuestionarios de dispositivo [2]

Una vez seleccionado el cuestionario, se mostrarán las cuestiones que el técnico deberá completar, las cuestiones con el texto rojo son obligatorias, y el usuario deberá completarlas para guardar el cuestionario.

Se avisará al usuario si trata de salir del cuestionario sin salvar las nuevas respuestas que haya completado.

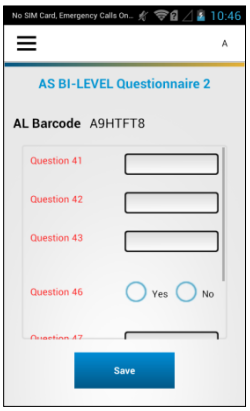

Figura 70: Cuestionario de dispositivo [1]

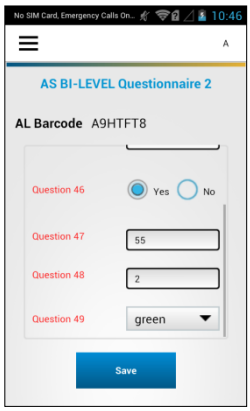

Figura 71: Cuestionario de dispositivo [2]

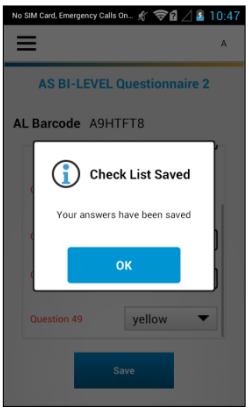

Figura 72: Cuestionario guardado

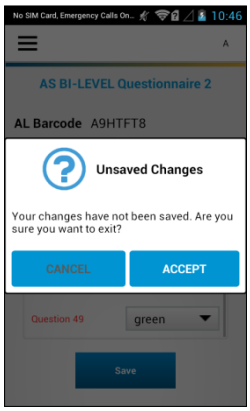

Figura 73: Cuestionario sin guardar

Una vez guardadas las cuestiones, el cuestionario aparecerá como editado para el código de barras asociado.

Para ciertos cuestionarios, es necesario añadir una línea especial a la lista de líneas de la DN al complementar estos.

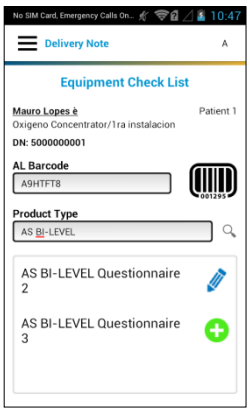

Figura 74: Cuestionario editado

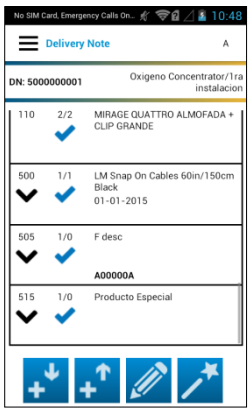

Figura 75: Línea especial

4.1.1.8.5. Notas del técnico

El usuario podrá añadir notas a la DN. Se avisará al usuario si puede perder los cambios no guardados.

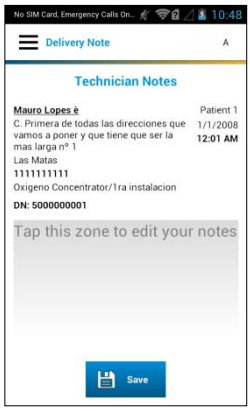

Figura 76: Notas del técnico

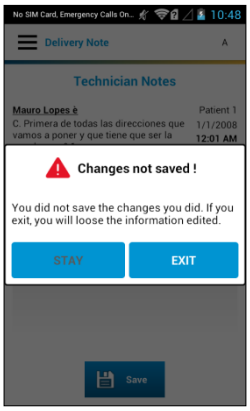

Figura 77: Nota sin guardar

4.1.1.8.6. Stock del paciente

El técnico podrá añadir y eliminar productos al stock del cliente, para ello deberá indicar el producto, y dependiendo de éste, el código de barras y el número de lote.

Para introducir el producto el usuario podrá valerse de un filtro igual que el explicado en detalles de entrega (véase 4.1.1.8.1).

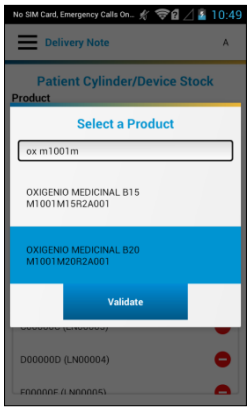

Figura 78: Selección de producto para stock

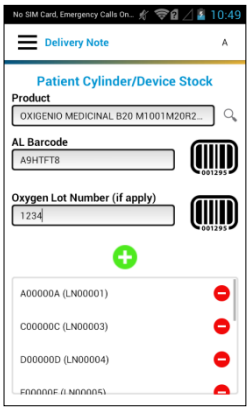

Figura 79: Añadir producto a stock [1]

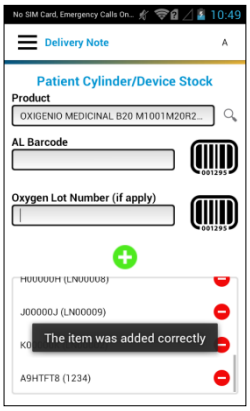

Figura 80: Añadir producto a stock [2]

4.1.1.8.7. Foto de seguridad

El técnico tendrá la posibilidad de tomar un número máximo de fotos (determinado en la configuración de servidor), por motivos de seguridad (véase 5.4.2) además de añadir comentarios a dichas fotos.

Del mismo modo al que se describió anteriormente, el sistema avisará de los posibles cambios perdidos.

Figura 81: Foto de seguridad [1]

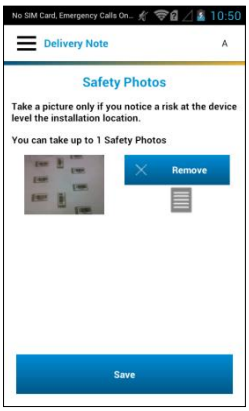

Figura 82: Foto de seguridad [2]

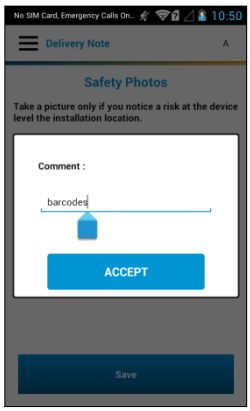

Figura 83: Foto de seguridad (comentario)

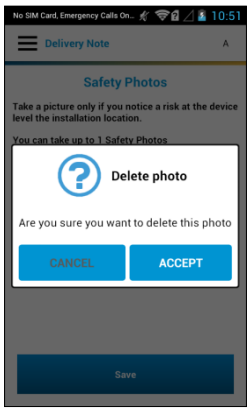

Figura 84: Eliminar foto

4.1.1.8.8. Transferencia de líquido

Transferencia de líquido: En este apartado el técnico podrá documentar una transferencia de líquido.

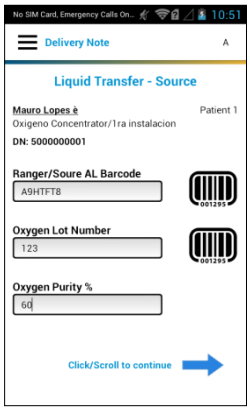

Figura 85: Origen de líquido

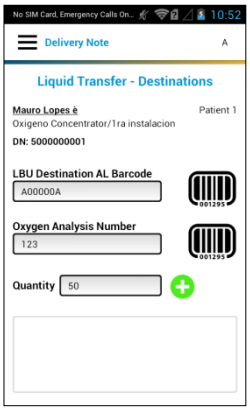

Figura 86: Destino de líquido

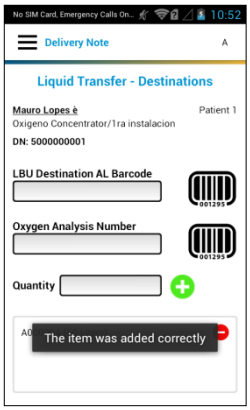

Figura 87: Transferencia

4.1.1.8.9. Posición

En esta ventana el usuario podrá capturar la posición en la que se encuentra (debe tomarla en algunos casos a la puerta del domicilio del paciente antes de firmar la DN).

Si el GPS no está activado, se mostrará un diálogo permitiendo activarlo.

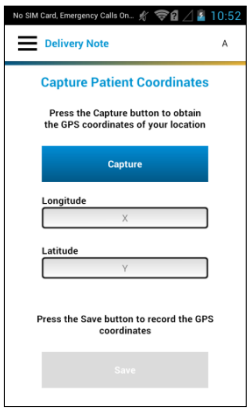

Figura 88: Capturar GPS

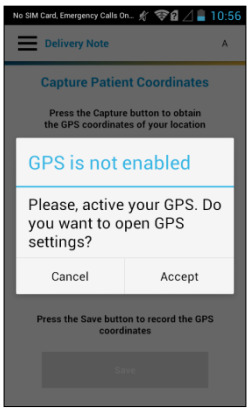

Figura 89: GPS deshabilitado

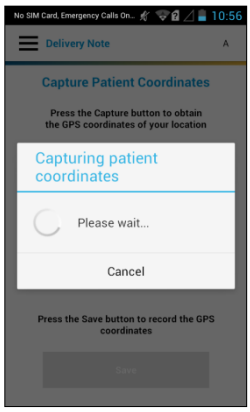

Figura 90: Capturando coordenadas

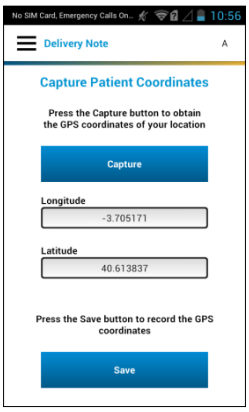

Figura 91: Coordenadas capturadas

4.1.1.8.10. Firmar y cerrar

En esta ventana se gestionará el proceso de firmado de DN y generación de eDN (véase Anexo A).

Durante este proceso el usuario podrá tomar una foto de la DN firmada en papel y llenar los campos de información del signatario. Además de informar si la firma del paciente es posible.

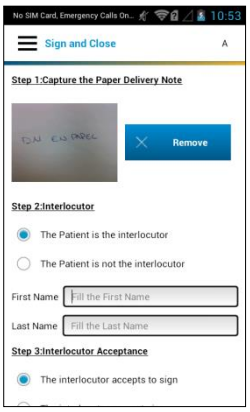

Figura 92: Firma y cierre [1]

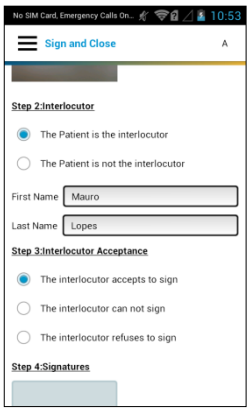

Figura 93: Firma y cierre [2]

El técnico podrá también capturar tanto su firma digital, como la del signatario (si decide firmar).

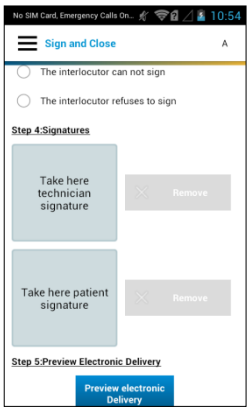

Figura 94: Firma y cierre [3]

Figura 95: Captura de firma

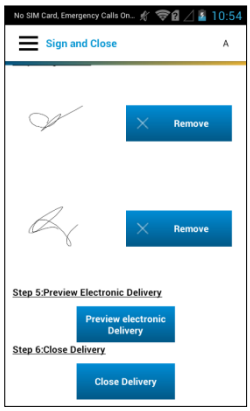

Figura 96: Firma y cierre [4]

Durante este proceso el técnico podrá mostrar una vista previa del documento al signatario, y tras completar la información, si éste está conforme, cerrar la DN y firmar el documento digital con el certificado de AL.

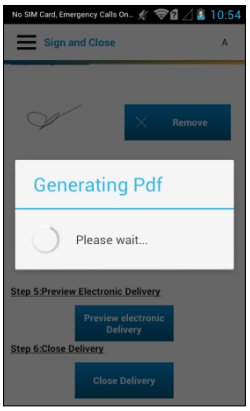

Figura 97: Creando eDN

Figura 98: eDN

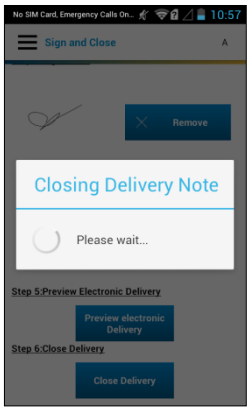

Figura 99: Cerrando DN

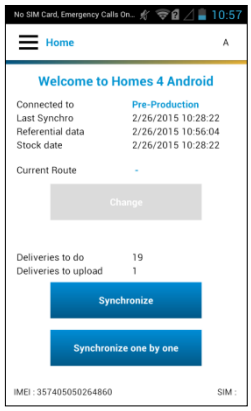

Figura 100: DN cerrada

4.1.2. Servidor HOMES

El servidor central HOMES está desarrollado por una tercera parte, la cual proporciona los WebServices necesarios para la comunicación con el mismo.

4.1.3. Base de datos interna

La aplicación cliente cuenta con una base de datos local gestionada mediante SQLite que permite gestionar las rutas, visitas y productos del servidor, así como almacenar los datos de la aplicación.

Caben destacar varias tablas especialmente relevantes (véase Anexo B):

- HM_SHIPMENT: Esta tabla almacena las rutas del servidor, está asociada con la tabla HM_USER que almacena los usuarios de HOMES, es decir los técnicos, y con la tabla HM_DELIVERY_NOTE, que almacena las visitas.
- HM_DELIVERY_NOTE: Esta tabla almacena las visitas, además de con la ruta, se relaciona con la tabla HM_DELIVERY_NOTE_DETAIL que contiene las líneas de detalle de cada visita.

En la base de datos también se pueden encontrar distintas tablas no relacionadas directamente (véase Anexo B figura 128).

5. DESARROLLO

En este apartado se documentará el proceso de desarrollo del proyecto HOMES 4.

5.1. Base de datos

Pese a que el modelo de la base de datos existía previamente en la aplicación, ha sido necesario migrarlo al sistema de gestión de bases de dato SQLite, incluido por defecto en Android.

Además de la migración se han incluido nuevos campos y tablas al modelo de datos.

Para lograr la compatibilidad del sistema con una base de datos SQLite ha sido necesario la creación de una implementación específica para SQLite de la interfaz de provider en la capa de datos (véase Anexo D.2.2).

En esta implementación se transforman los valores .NET en valores aceptados por SQLite, por ejemplo booleanos se transforman en 0 o 1. Adicionalmente se añade seguridad frente a las inyecciones SQL.

Adicionalmente, algunas consultas fallaban al ejecutarse en SQLite, de manera que ha sido necesario adaptarlas a SQLite.

5.1.1. Tecnologías utilizadas

Para facilitar la transformación de algunas consultas y detectar sus fallos se ha utilizado la herramienta SQLiteStudio.

Por otra parte, para añadir elementos a la base de datos del servidor (la cual no entra dentro del alcance de este TFG) con objeto de depurar la aplicación, se ha utilizado SQLServer 2008.

5.2. Aplicación Android

Este apartado pretende explicar el proceso de desarrollo de la aplicación cliente Android (Homes.Android).

Debido a la confidencialidad del código, y a los aspectos poco relevantes de ciertas funcionalidades, se detallarán únicamente aquellas cuya complejidad lo requiera.

5.2.1. Tecnologías utilizadas

Para codificar el cliente Homes.Android, se ha utilizado la herramienta Xamarin.Android. Esta herramienta permite la utilización de las APIs de google mediante el lenguaje de programación C# además de permitir el uso del framework .NET.

Esto es crucial, ya que todo el back office del sistema utiliza este framework (véase 3.2).

Adicionalmente para el desarrollo de layouts (XML) se ha utilizado el entorno Android Studio, que proporciona consejos a la hora de generar layouts.

Para la generación y firma digital de PDF se ha requerido el uso de la herramienta iTextSharp [(8)].

Para el manejo del escáner lineal del terminal se ha utilizado la biblioteca EMDK proporcionada por Motorola Solutions [(9)].

5.2.2. Creación del esqueleto de la aplicación:

La primera etapa de desarrollo se concentró exclusivamente en la creación de la interfaz gráfica de la aplicación en Android.

En esta etapa se siguió un proceso iterativo en el que, a partir del diseño proporcionado por el cliente, se mantuvo un intercambio de implementaciones y correcciones hasta obtener el esqueleto final.

Debido a esto, no ha sido posible la libertad de cambiar determinados diseños que no son habituales en Android, lo que ha supuesto un desafío a la hora de crear ciertas vistas poco habituales y, en algunos casos, con excesiva funcionalidad.

A continuación se documenta el proceso de desarrollo de los puntos más relevantes de la interfaz.

5.2.2.1 Organización de actividades

Las clases de la actividad se organizan siguiendo el siguiente esquema:

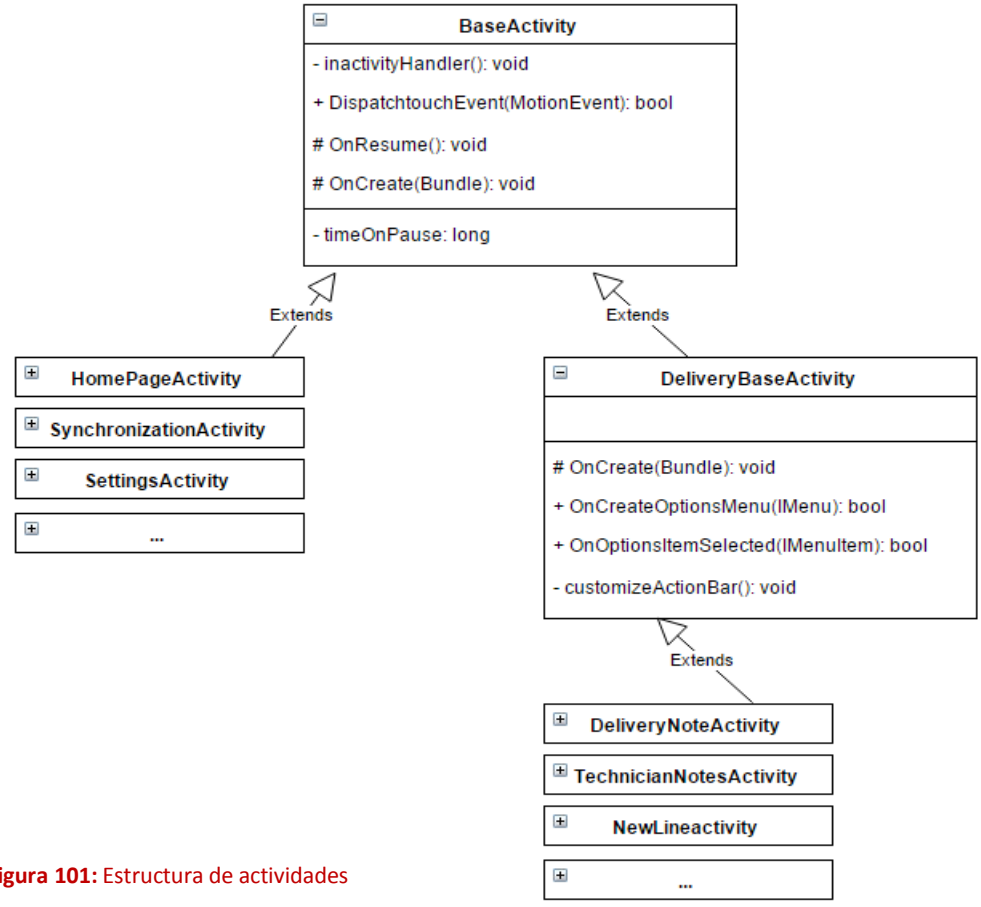

Figura 101: Estructura de actividades

- LoginActivity: Clase sin historia, se destruye en cuanto pasa a segundo plano.
- BaseActivity: Clase base de la aplicación de la que parten el resto de clases excepto LoginActivity. Utiliza los métodos DispatchTouchEvent y inactivityHandler para controlar el bloqueo por inactividad del terminal [véase 3.3.1.1, RF9].
- DeliveryBaseActivity: Clase que hereda de BaseActivity y que a su vez sirve como clase base para todas las actividades relacionadas con las DN.

5.2.2.2 Adaptadores

Los adaptadores permiten el acceso a los datos de los elementos de una lista, y además permiten definir las vistas de cada elemento contenido dentro de una vista superior [(10)].

Dentro de la aplicación se han utilizado adaptadores para cumplir los siguientes objetivos:

- Listas personalizadas con botones funcionales o distribución especial de sus elementos (véanse figuras 51, 75, apartado 4.1.1)
- Listas con filtrado de elementos (véase figura 84, apartado 4.1.1.8.6); en este caso el adapter permite acceder a los datos, filtrarlos y devolvérselos a la vista.
- Personalización de otros grupos de vistas, como paneles laterales, Spinners o ViewPagers.

5.2.2.3 Layouts y diseño de pantallas

Al programar interfaces visuales en Android, se permite definir los elementos gráficos y su distribución mediante ficheros XML.

Para Homes.Android se ha seguido este procedimiento, definiendo distintos layouts para cada actividad o notificación que quiera mostrarse en la pantalla del terminal.

A la hora de diseñar los elementos gráficos de la aplicación, hay que tener en cuenta que los terminales en los que se ejecutará la aplicación pueden tener distintos tamaños de pantalla y distintas resoluciones. A pesar de que la aplicación tiene un dispositivo objetivo, es una buena práctica diseñar una interfaz gráfica que se adapte a diferentes pantallas.

Pese a que las etiquetas XML que proporciona Android para el diseño de layouts permiten definir el tamaño y posición de los elementos de forma relativa a la pantalla, al añadir recursos de imagen o el tamaño de las fuentes, es recomendable tener en cuenta los DPI (Dots per inch, puntos por pulgada) de cada terminal.

Android permite definir los recursos de la aplicación para diferentes calificadores, como pueden ser los DPI o tamaño de la pantalla o el idioma del terminal. Para diseñar una interfaz capaz de adaptarse a los distintos terminales del mercado se han añadido diferentes recursos de imagen y de fuente para cada franja de DPI, siguiendo la guía de diseño para el soporte de diferentes pantallas de Android [(11).]

En la guía de diseño se propone la siguiente tabla de equivalencias:

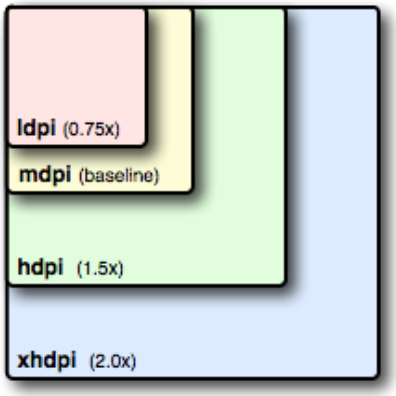

Figura 102: Relación de densidades

Por comodidad, para obtener las equivalencias de los distintos recursos se ha utilizado la herramienta web (http://coh.io/adpi/), tras comprobar que escalaba correctamente, según la escala proporcionada por la guía.

Adicionalmente, es posible personalizar el aspecto y respuesta visual de los elementos que conforman estos layouts. Para ello se crean ficheros XML de tipo drawable [(12)], que se asignarán a la etiqueta correspondiente dentro del elemento que se quiera personalizar.

5.2.2.4 Diálogos

En las aplicaciones de Android, los diálogos permiten mostrar al usuario una notificación sobre una ventana y recibir su respuesta a dicha información.

Por exigencias del cliente, todos los diálogos de la aplicación debían mostrarse personalizados, de acuerdo a un diseño determinado.

Dentro del desarrollo de los diferentes tipos de diálogos necesarios para la aplicación, se pueden distinguir los siguientes grupos:

- Diálogos simples.
- Diálogos con filtro.

5.2.2.4.1 Diálogos simples

Dentro del grupo de los diálogos simples podemos englobar diferentes personalizaciones de diálogos de confirmación (muestran un mensaje y esperan la confirmación del usuario) o de elección (muestran un mensaje y esperan la elección del usuario).

Estos diálogos se pueden personalizar de manera sencilla, para realizarlo se ha utilizado un fichero XML para especificar la forma, color y contorno de la ventana, y un fichero XML para definir el layout con el contenido de la vista (Cada tipo de diálogo tiene un layout con elementos distribuidos según su necesidad).

Todos los diálogos de este grupo se crean mediante un objeto Dialog[(13)] al que se carga el fondo y el layout correspondiente.

5.2.2.4.2 Diálogos con filtro

Este tipo de diálogos suponen una mayor complejidad, debido a la necesidad de incorporarles una lista y un elemento que permita filtrarla.

Para su desarrollo es necesario crear una clase que extienda de “dialog”, y diseñar un adapter diferente para cada tipo de diálogo, encargado de aplicar la lógica de filtrado sobre los elementos de la lista (Al ser elementos distintos según el tipo de diálogo ha sido necesario crear distintos adaptadores).

Todos los adapter de filtro heredan de una clase abstracta base, con el motivo de lograr una mayor abstracción y reducir el número de métodos necesarios dentro de la clase del diálogo.

Estos adapter deben sobrescribir los métodos de la clase base, necesarios para implementar la interfaz IFilterable [(14)], de esta forma cada adapter permite filtrar diferentes listas con diferentes criterios.

Para utilizar estos diálogos, el proyecto cuenta con una clase estática que permite la creación de los diferentes objetos de dialogo personalizados.

5.2.2.5 ViewPager

Este tipo de vista permite la transición dentro de una misma actividad de los fragments que la componen mediante desplazamiento vertical.

Un fragment es un fragmento o porción de la interfaz de usuario de una actividad.

Se han utilizado ViewPagers en diferentes secciones de la aplicación (véanse 4.1.1.8.1 secuencia editar línea ó 4.1.1.8.8) para dar fluidez al manejo de la misma.

5.2.3. Comunicando la aplicación con el servidor

La comunicación entre el servidor y la aplicación, se realiza mediante el intercambio de tramas CSV (véase Anexo D.2.1) mediante llamadas HTTPS (véase Anexo D.4).

El servidor mantiene una serie de WebServices a los que accederá la aplicación para el intercambio de diferentes datos.

Durante el desarrollo de la aplicación ha sido necesario integrar diversos WebServices disponibles en el servidor, a continuación se explica cómo se ha integrado la funcionalidad que proporcionan dichos WebServices en el sistema.

Para integrar cualquier WebService en el sistema hay que seguir los siguientes pasos:

- Añadir una clase para cada servicio dentro de la colección de servicios mantenida en la capa HOMES (véase Anexo D.1.2), también deben añadirse los métodos necesarios para transformar las trama CSV en datos utilizables por la aplicación.
- Añadir los métodos que se encargarán de las llamadas a las clases anteriores en el BF encargado de los WebServices, WebServiceBF.
- Añadir los métodos que se encargarán de las llamadas a WebServiceBF en los FBF adecuados (dependiendo de la operación que se realice).

5.2.3.1 Búsqueda de identificador de producto por código de barras

En la versión previa de Homes, cuando un técnico añade un dispositivo no esperado (el dispositivo no se encuentra dentro del stock), se añadía una línea de detalle vacía a la DN.

Para ello el técnico añadía un dispositivo, solo especificando el código de barras. Cuando los datos se sincronizaban con el servidor se comprobaba el producto asociado a este código de barras.

Las razones de esta implementación eran las siguientes:

- No se descargaba la lista de dispositivos completa.
- No existía una conexión 3G/GPRS.

Sin embargo, las necesidades de Italia obligan a que en la eDN (véase 3.3.1.2, RF21, RF29) aparezcan todos los detalles de los dispositivos entregados o recogidos.

Para gestionarlo, se dispone de un WebService que dado un determinado código de barras, devolverá el código de producto asociado (BizID).

5.2.4. Configuración de la cámara del dispositivo

Las necesidades del cliente indican que la aplicación no debe permitir al usuario cambiar la configuración de la cámara, dicha configuración se descargará del servidor central pudiendo ser diferente para cada usuario.

El método más habitual de controlar la cámara en una aplicación Android es mediante un Intent que invoque a la cámara nativa de Android. Este método es simple, pero está limitado en cuanto a personalización.

En nuestro caso este método no es viable debido a que, aunque sea posible configurar los parámetros de la aplicación nativa de la cámara en tiempo de ejecución, no es posible evitar que el usuario realice cambios cuando la cámara de Android se despliegue. Es necesario que la configuración de la cámara sea transparente al usuario en todo momento.

Debido a esto, la solución más apropiada es la construcción de una cámara propia dentro de la aplicación, para lograr esto Android cuenta con las siguientes clases en su API:

- *Camera*: Esta clase permite controlar el hardware de la cámara[(15)] (clase obsoleta).
- *Camera.Parameters*: Esta clase permite configurar los parámetros de la cámara [(16)] (clase obsoleta).
- *Camera 2*: Reemplaza la clase camera, a partir del API 21 [(17)].
- *SurfaceView*: Esta clase permite presentar al usuario una vista previa de la cámara [(18)].
- *ISurfaceHolderCallback*: Interfaz de Xamarin equivalente a SurfaceHolderCallback de Android, permite, al implementarla capturar cambios en la vista SurfaceView [(19)].

– *IPictureCallback*: Interfaz de Xamarin equivalente a PictureCallback de Android, permite, al implementarla capturar la foto capturada por la cámara.

La aplicación debe ser compatible con cualquier versión de Android a partir de Android 4.0. La clase camera 2 no resuelve esta necesidad, de manera que se ha utilizado la clase obsoleta Camera, en conjunto con las clases Camera.Parameters y SurfaceView.

Contando con estas clases, para la creación de una cámara que se adapte a las necesidades se han creado las siguientes clases dentro del proyecto:

* *CameraPreview*: Clase que hereda de SurfaceView y que implementa ISurfaceHolderCallback
* *CameraActivity*: Actividad dentro de la aplicación desde la que se lanzará la cámara, asignándole a la vista de la actividad la vista previa (CameraPreview) y los parámetros (Camera.Parameters) con la configuración descargada del servidor (calidad de imagen, resolución, flash…). Esta clase obtendrá la fotografía resultante implementando el método OnPictureTaken de IPictureCallback.

5.2.5. Generación de PDF

A continuación se documentará el proceso de generación de DN digital mediante la herramienta iTextSharp (véase 3.3.1.2 RF21).

Con el objetivo de reducir la cantidad de papel, la compañía cliente requiere la implantación de DN firmada y en formato digital, de modo que ésta sirva de comprobante de visita.

Para la implementación de este proceso, será necesario seguir un flujo de ejecución (véase Anexo A).

El uso de la herramienta iTextSharp se puede observar con mayor detalle en el anexo E: Manejo de iTextSharp.

5.2.6. Manejo del lector lineal

A continuación se describe la implementación del requisito funcional RF11 (véase 3.3.1.1). Este requisito exige la utilización del hardware del terminal TC55 para la lectura de códigos de barra.

La implementación de esta tarea puede abordarse desde dos enfoques, ambos contemplados en detalle en el Anexo F: Abordando el manejo del lector lineal.

5.2.7. Uso de funcionalidades propias de C#

En algunos puntos de la aplicación, ha resultado útil poder utilizar ciertas funcionalidades de C# no existentes en java, a continuación se desarrollan algunas de ellas.

5.2.7.1 Delegados

Los delegados permiten pasar referencias a métodos como argumentos a otros métodos. A continuación veremos un ejemplo donde fueron de gran utilidad:

A la hora de realizar el autofill (véase 4.1.1.8.1) surge el siguiente problema:

- Existe una funcionalidad del back office que permite iterar en cada línea de detalle y autocompletarla, sin embargo un nuevo requerimiento de última hora exige que en determinadas ocasiones el usuario confirme la lectura de un aviso, bloqueando así la ejecución del bucle.
- Para solucionarlo es posible utilizar los delegados para pasar a la fachada un método encargado de bloquear la ejecución del bucle hasta la confirmación del usuario.

5.2.7.2 LINQ

LINQ es un componente de .NET que permite aplicar consultas sobre colecciones de datos como listas o arrays. En este proyecto ha resultado de extrema utilidad, y ha servido para ordenar y buscar elementos en gran cantidad de diferentes colecciones de datos.

5.2.7.3 ASYNC

Se ha utilizado async [(20)] para gestionar procesamiento en segundo plano, con el objetivo de evitar el bloqueo de la interfaz gráfica del terminal y el correspondiente diálogo informando de que la aplicación no responde [(21)].

6. PRUEBAS

En este apartado se pretende documentar las pruebas realizadas sobre el sistema.

6.1. Pruebas de interfaz gráfica

En este apartado se detallan las pruebas realizadas sobre la interfaz gráfica del sistema.

6.1.1. Flujo de la aplicación

Se comprueba el correcto funcionamiento del flujo de actividades. Para ello se han explorado todos los posibles caminos y sus retornos.

6.1.2. Campos de entrada

Se ha comprobado el correcto funcionamiento de la aplicación para los campos de entrada. Para ello se han realizado las siguientes comprobaciones:

- Casos base: La aplicación captura y procesa bien las entradas esperadas en los campos.
- Casos extremos: Se han probado casos de entrada máximos y mínimos. La aplicación responde correctamente en ambos casos para todos los campos de entrada, salvo para los filtros donde se encontró un bug al escribir o borrar rápido. Este problema estaba relacionado en la forma en la que se transformaban los objetos en objetos java, la solución requería hacer un casting a Java.Lang.String del string .Net al llamar al filtro [(22)].

  En la mayoría de casos se define un máximo de caracteres en las etiquetas XML del layout, o como en el caso de los códigos de barras existe un método que comprueba su validez antes de poder avanzar.
- Tipo incorrecto: En la mayoría de casos, la API de Android permite utilizar el teclado evitando entradas incorrectas. Únicamente se permite el teclado completo en los códigos de barras, notas y respuestas de cuestionarios, en el primer caso se comprueba la validez del campo, y en el resto se permite cualquier formato de entrada.
- Lector de códigos de barra:
  * Se comprueba la lectura en todas las pantallas en las que esté involucrado.
  * Se comprueba que la aplicación responde correctamente cuando el terminal no dispone de lector lineal de código de barras.

6.1.3. Pruebas de resolución y tamaño de pantalla

Para probar el comportamiento de la interfaz en dispositivos con pantallas de diferente resolución y tamaño de pantalla, se ha repetido la prueba de flujo (véase 6.1.1) en los siguientes terminales:

Tabla 4: TIPOS DE PANTALLAS PROBADOS

TERMINAL	RESOLUCIÓN (pixeles)	TAMAÑO DE PANTALLA (pulgadas)
- Motorola TC55	480 x 800	4,3
- Kazam Thunder2 4.5L	480 x 854	4,5
- ToughShield R500+	480 x 800	3,77
- Sony Xperia S	720 x 1280	4,3
- Samsung Galaxy Tab 3	600 x 1024	7
- Galaxy One Note 2	720 x 1280	5,5
- Motorola Moto G	720 x 1280	4,5
- Nexus 7	1200 x 1920	7

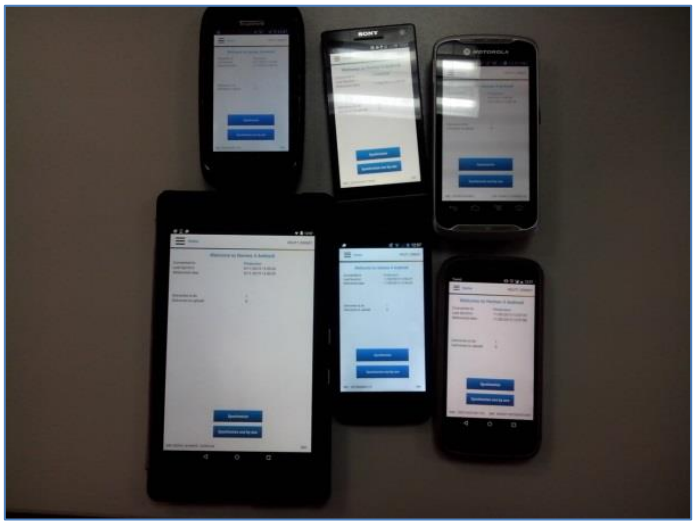

Figura 103: HMS en diferentes pantallas

6.1.4. Pruebas de versión de S.O.

Se ha probado el funcionamiento de la aplicación en las siguientes versiones de Android:

Tabla 5: VERSIONES DE ANDROID PROBADAS	
TERMINAL	VERSIÓN
- Motorola TC55	4.1.2 y 4.4
- Kazam Thunder2 4.5L	4.3
- ToughShield R500+	4.1.2
- Sony Xperia S	4.0.4
- Samsung Galaxy Tab 3	4.2.2
- Galaxy One Note 2	4.3
- Motorola Moto G	5.0.2
- Nexus 7	5.1

6.2. Pruebas de base de datos local

Para poder acceder a la base de datos de la aplicación se ha determinado un directorio temporal para la base de datos durante las pruebas.

Se ha comprobado la persistencia de los datos introducidos desde la aplicación, y la correcta recuperación de datos desde la aplicación.

6.3. Pruebas de base de datos central

Para probar el correcto envío de datos al servidor por medio de HTTPS se ha configurado un servidor en una de las máquinas de la oficina, replicando la base de datos del servidor central.

A partir de dicho servidor se ha comprobado la transferencia de datos entre la aplicación y el servidor.

7. CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se exponen las conclusiones personales, así como el posible trabajo futuro del proyecto.

La evolución tecnológica se ha cumplido satisfactoriamente, resultando un éxito la portabilidad de la aplicación a Android, así como todas las funcionalidades añadidas descritas en los requisitos funcionales.

Una vez comprendido el back office de la aplicación fue un reto introducirlo con las particularidades de un sistema Android.

La mayor dificultad se presentó durante el análisis del back office de dicha aplicación, así como el diseño de determinadas pantallas, que no estaban pensadas para un entorno tradicional de las pantallas Android.

Durante el tiempo dedicado a este Trabajo Fin de Grado, además de aprender a desarrollar un proyecto completo, pasando por todas sus fases, he adquirido nuevos conocimientos muy interesantes y ampliado muchos otros relacionados con C#, Android, .NET, SQL, Modelo fachadas y plataformas cruzadas, así como nociones necesarias para la migración y evolución realizada.

Respecto al trabajo futuro, el sistema estará abierto a una etapa de mantenimiento y posiblemente a cambios de versiones.

Además igual que se ha realizado una migración a Android mediante Xamarin, se podrían hacer migraciones tanto a iOS como a Windows Phone si, en un escenario poco probable, dominaran el mercado de terminales industriales.

BIBLIOGRAFÍA

1. **Microsoft.** Microsoft Support Lifecycle. *https://support.microsoft.com/en-us/lifecycle/search/default.aspx?sort=PN&alpha=Windows%20Mobile%206&Filter=FilterNO.* [En línea] Microsoft, 2015. https://support.microsoft.com/en-us/lifecycle/search/default.aspx?sort=PN&alpha=Windows%20Mobile%206&Filter=FilterNO.

2. **MSDNBlogs.** El futuro de Windows Mobile 6.x y Windows CE. *blogs.msdn.com.* [En línea] 13 de 07 de 2011. http://blogs.msdn.com/b/oaviles/archive/2011/07/13/el-futuro-de-windows-mobile-6-x-y-windows-ce-handheld-devices.aspx.

3. **IDC.** Smatphone market share. *http://www.idc.com/prodserv/smartphone-market-share.jsp.* [En línea] Smartphone Vendor Market Share, 2014. http://www.idc.com/prodserv/smartphone-market-share.jsp.

4. **Droid-Life.** Android Distribution. *droid-life.* [En línea] 04 de 2015. http://www.droid-life.com/2015/03/02/android-distribution-updated-for-march-2015-lollipop-now-at-3-3/.

5. **Microsoft.** Silverlight. *http://es.wikipedia.org/wiki/Microsoft_Silverlight.* [En línea] 2015. https://www.microsoft.com/silverlight/what-is-silverlight/.

6. **Wikipedia.** Xamarin. *http://es.wikipedia.org/wiki/Xamarin.* [En línea] 2015. http://en.wikipedia.org/wiki/Xamarin.

7. **Xamarin.** Xamarin platform. *http://xamarin.com/.* [En línea] 2015. http://xamarin.com/platform.

8. **iTextSharp.** iTextSharp en sourceforge. [En línea] http://sourceforge.net/projects/itextsharp/.

9. **Motorola Solutions.** EMDK. [En línea] https://developer.motorolasolutions.com/docs/DOC-2281.

10. **androidforbegginers.** android adapters introduction. [En línea] http://androidforbeginners.blogspot.com.es/2010/01/android-adapters-introduction.html.

11. **developer.android.** Supporting Multiple Screens. [En línea] http://developer.android.com/guide/practices/screens_support.html.

12. —. Drawable Resources. [En línea] http://developer.android.com/guide/topics/resources/drawable-resource.html.

13. —. Dialogs. [En línea] http://developer.android.com/guide/topics/ui/dialogs.html.

14. **androidapi.xamarin.** Android.Widget.IFilterable. [En línea] http://androidapi.xamarin.com/index.aspx?link=T%3AAndroid.Widget.IFilterable.

15. **developer.android.** Camera. [En línea] http://developer.android.com/reference/android/hardware/Camera.html.

16. —. Camera.Parameters. [En línea]
http://developer.android.com/reference/android/hardware/Camera.Parameters.html.

17. —. android.hardware.camera2. [En línea]
https://developer.android.com/reference/android/hardware/camera2/package-summary.html.

18. —. SurfaceView. [En línea]
http://developer.android.com/reference/android/view/SurfaceView.html.

19. **androidapi.xamarin.** Android.Views.ISurfaceHolderCallback. [En línea]
http://androidapi.xamarin.com/index.aspx?link=T%3AAndroid.Views.ISurfaceHolderCallback.

20. **msdn.microsoft.** Programación asincrónica con Async y Await (C# y Visual Basic). [En línea] https://msdn.microsoft.com/es-es/library/hh191443.aspx.

21. **developer.android.** Processes and Threads. [En línea]
http://developer.android.com/guide/components/processes-and-threads.html.

22. **Bugzilla.xamarin.** Bug 25995 - SIGABRT using Filter. [En línea]
https://bugzilla.xamarin.com/show_bug.cgi?id=25995.

23. **Eric Freeman, Kathy Sierra, Bert Bates, Elisabeth Robson.** *Head First Design Patterns.* s.l. : Paperback, 2004.

24. **ietf.** Common Format and MIME Type for Comma-Separated Values (CSV) Files. [En línea] http://www.ietf.org/rfc/rfc4180.txt.

25. **Lowagie, Bruno.** *iText In Action.* Stamford : Manning Publications Co, 2011.

26. **api.itextpdf.** Class PdfTemplate. [En línea]
http://api.itextpdf.com/itext/com/itextpdf/text/pdf/PdfTemplate.html.

27. **developer.motorolasolutions.com.** Scan barcodes using DataWedge Intents in Xamarin Android. [En línea] https://developer.motorolasolutions.com/community/android/android-forums/android-blogs/blog/2014/11/06/scanning-barcodes-in-your-xamarin-android-app.

28. **andoridadpi.xamarin.** Android.App.Activity.OnNewIntent. [En línea]
http://androidapi.xamarin.com/index.aspx?link=M%3AAndroid.App.Activity.OnNewIntent%28Android.Content.Intent%29.

29. **docs.oracle.com.** Java Archive (JAR) Files. [En línea]
http://docs.oracle.com/javase/6/docs/technotes/guides/jar/index.html.

30. **developer.xamarin.** Binding a Java Library. [En línea]
http://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/binding_a_java_library_(.jar)/.

31. **developer.motorolasolutions.com.** Xamarin and the Zebra EMDK. [En línea]
https://developer.motorolasolutions.com/docs/DOC-2813.

32. **developer.xamarin.** API Metadata Reference. [En línea]
http://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/binding_a_java_library_%28.jar%29/api_metadata_reference/.

ANEXOS

Anexo A: Diagrama eDN

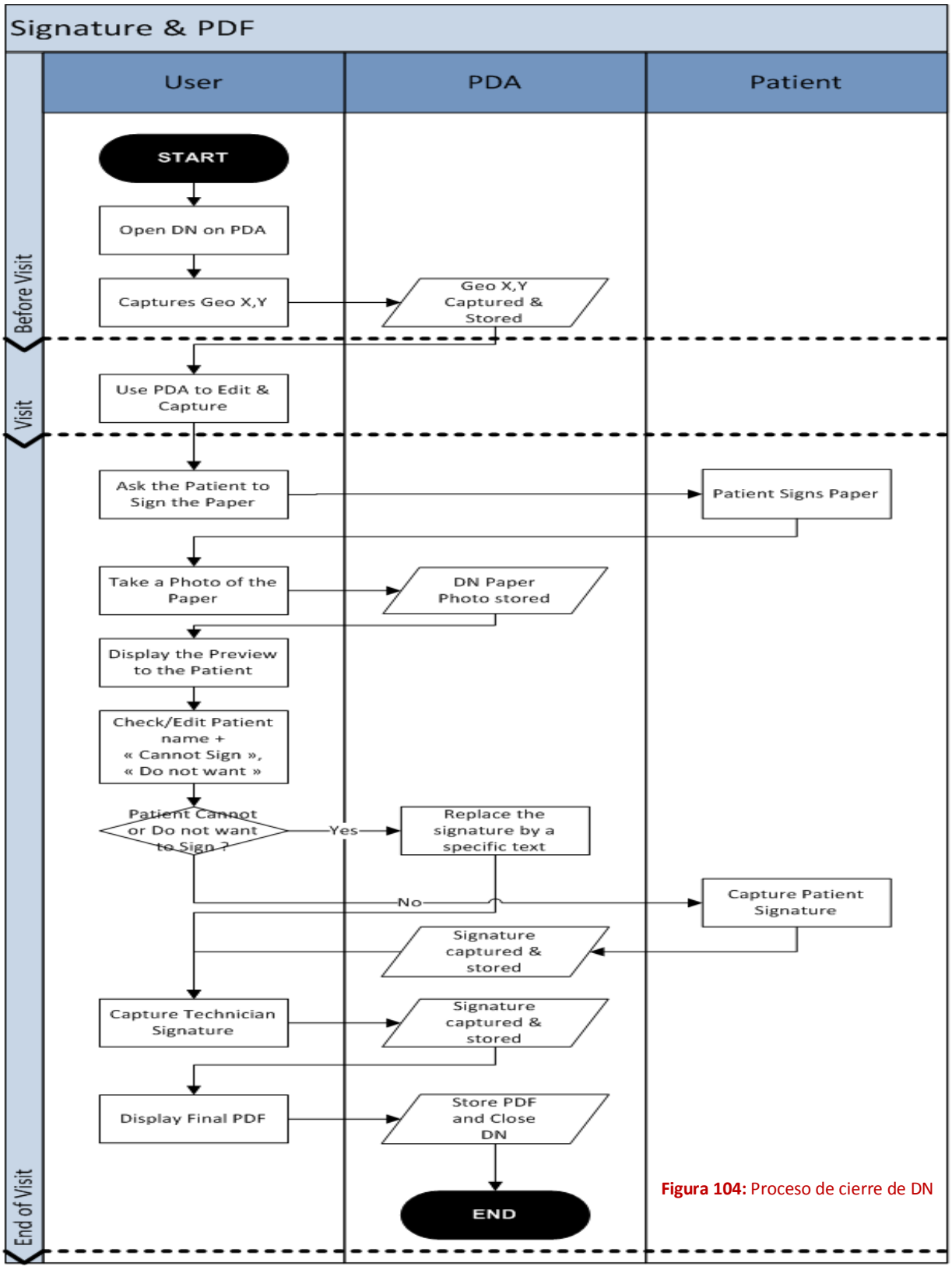

Figura 104: Proceso de cierre de DN

Anexo B: Tablas de base de datos local

Figura 105: Diagrama base de datos local

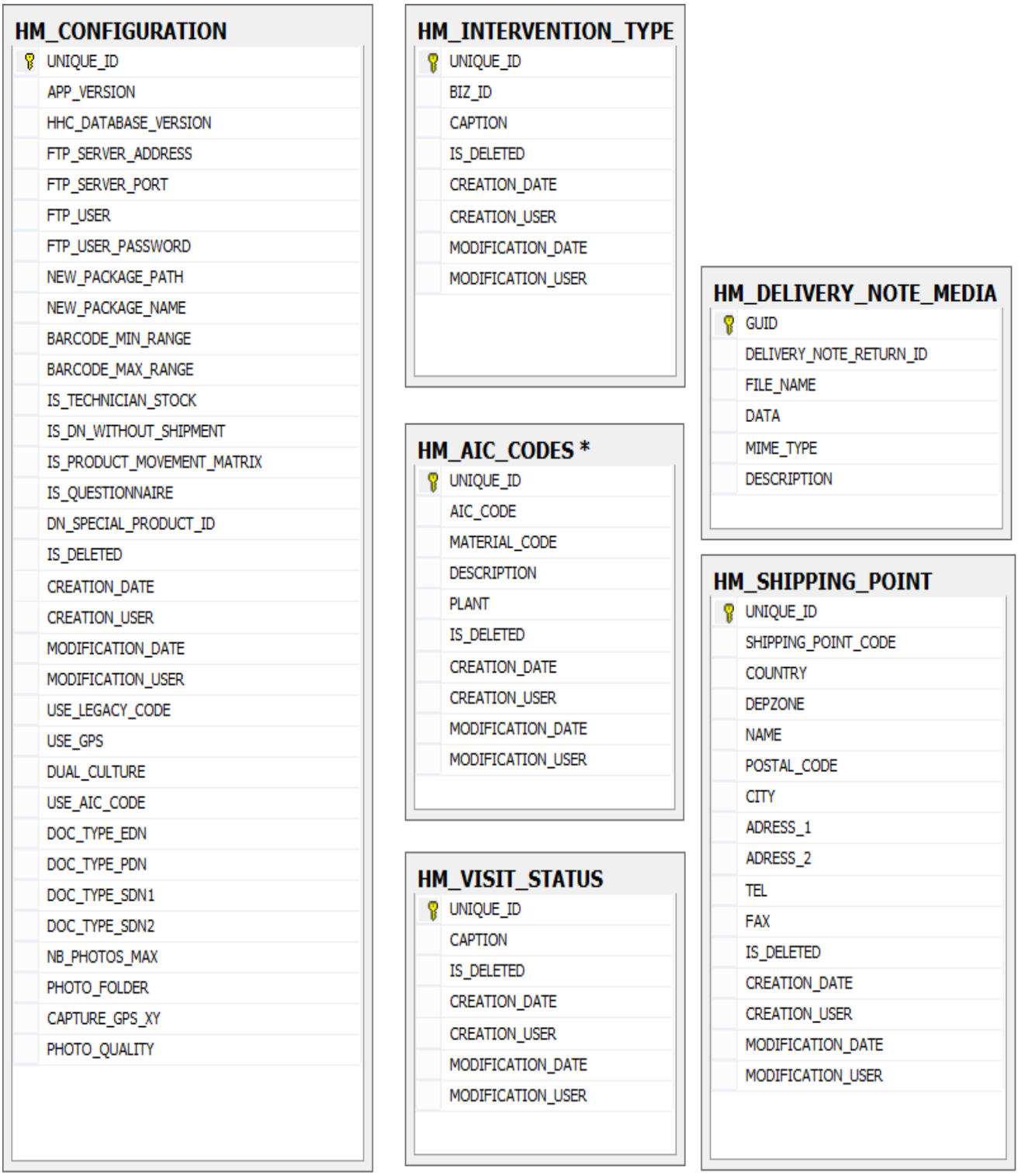

Figura 106: Tablas no relacionadas

Anexo C: Análisis de la aplicación previa

A continuación se detallarán las funcionalidades de cada apartado de la aplicación HMS 3.0, para ello se listarán las reglas existentes, y se mostrará el diagrama de secuencia y el flujo de pantallas de cada uno de ellos.

C.1. Sincronización

El técnico debe realizar el proceso de sincronización antes de comenzar su ruta para descargar los datos necesarios, y al finalizarla, para subir los datos completados al servidor.

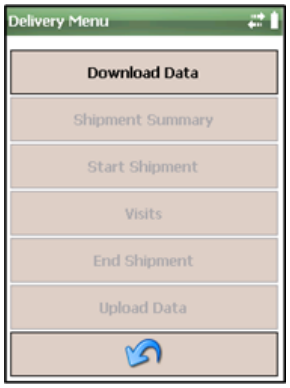

Figura 107: HMS3 Sincronización

C.2. Resumen

Esta funcionalidad permite al usuario revisar la lista de DNs, y la lista de líneas de detalle de cada una de ellas.

Una línea de detalle, se corresponde con la entrega o recogida de uno o varios productos dentro de una DN.

Además el usuario puede comprobar los productos disponibles en su stock.

Reglas:

Las siguientes pantallas eran accesibles únicamente si el usuario ha descargado previamente las DN y el stock en su terminal (véase C.1).

- Para el resumen de DNs:
  - El usuario puede mostrar la lista de DNs pertenecientes a su ruta.
  - Cada DN aparece en el orden definido por el servidor.
  - Se muestra el nombre del cliente asociado a cada DN.

- Desde esta pantalla, el técnico puede acceder a una DN y verla en detalle (acceder a sus líneas de detalle).
- Desde esta pantalla, el técnico puede acceder a un resumen de los productos disponibles en su stock (acceder al stock del vehículo).

➢ Para las líneas de detalle de una DN:

- En esta pantalla, el usuario puede ver una lista con las líneas de detalle asociadas a la DN contenedora, dichas líneas están formadas por:
  - Una descripción de un producto entregado o recibido.
  - Un número de línea único, múltiplo de 10 en caso de ser líneas estándar descargadas del servidor, y múltiplo de 5 en caso de ser líneas inesperadas.
- Cada línea de detalle aparece en el orden definido por el servidor.
- Desde esta pantalla el usuario puede volver a la lista de DNs.

➢ Para el stock de productos:

- En esta pantalla el usuario puede ver el resumen de todos los productos disponibles en su stock.
- Los productos aparecen ordenados por nombre.
- Para cada producto se muestra la siguiente información:
  - Descripción/nombre del producto.
  - Cantidad disponible del producto.
  - El código de barras o número de lote asociado al producto.

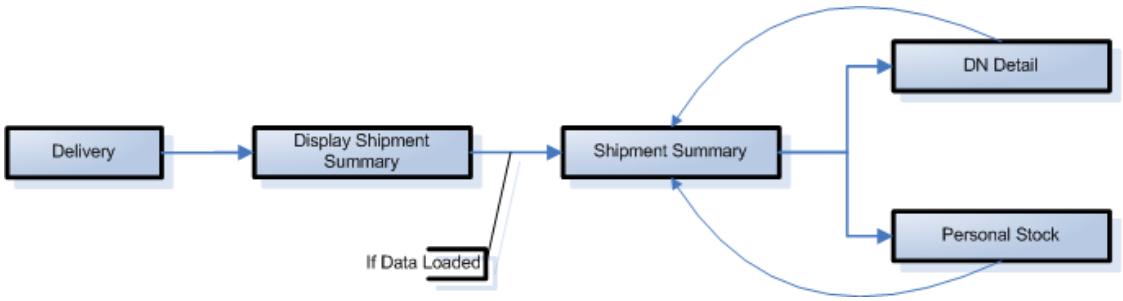

Figura 108: HMS 3 Esquema de resumen

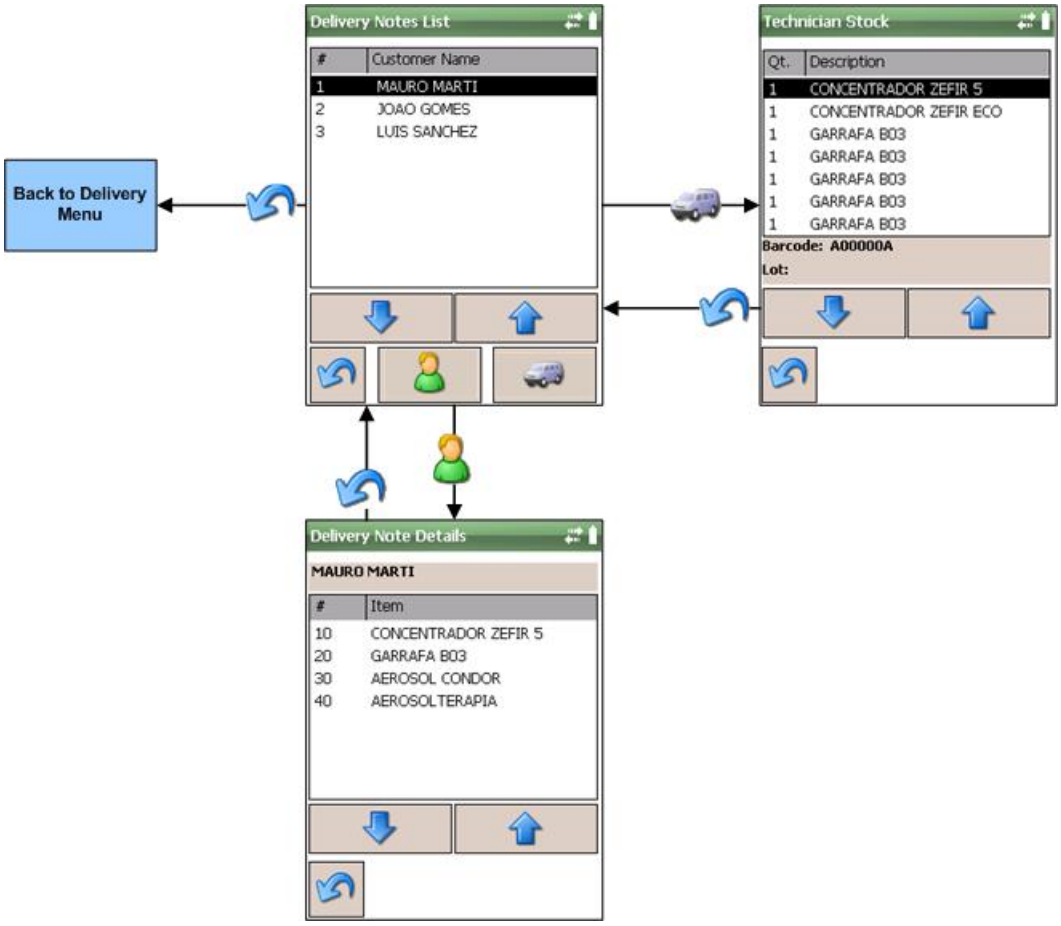

Figura 109: HMS3 Flujo de resumen

C.3. Inicio y finalización de rutas

Esta pantalla permite al usuario declarar una ruta como comenzada. El sistema captura la fecha y hora y permite al usuario introducir los kilómetros del vehículo.

Esta funcionalidad no se mantendrá en Homes.Android.

C.4. Visitas

Este apartado permite al usuario las siguientes acciones:

- Mostrar la lista de pacientes a visitar, en el orden indicado por el servidor.
- Mostrar el estado de la visita (Visitado /no visitado).
- Mostrar los detalles del paciente al ser seleccionado en la lista (código de paciente, dirección, teléfono…).
- Entrar en las líneas de detalle de la visita tras marcarla y pulsar el botón.

C.4.1. Detalles de visita

Cuando el usuario entra en los detalles de una visita, es capaz de realizar las siguientes acciones:

- Mostrar la lista de pacientes a visitar, en el orden indicado por el servidor.
- Acceder a las líneas de detalle de la DN
- Completar los formularios acerca de los dispositivos.
- Marcar la visita como imposible.

C.4.1.1 Productos

Durante la visita, el usuario podrá gestionar una serie de productos, que contendrán o no un número de lote y un código de barras.

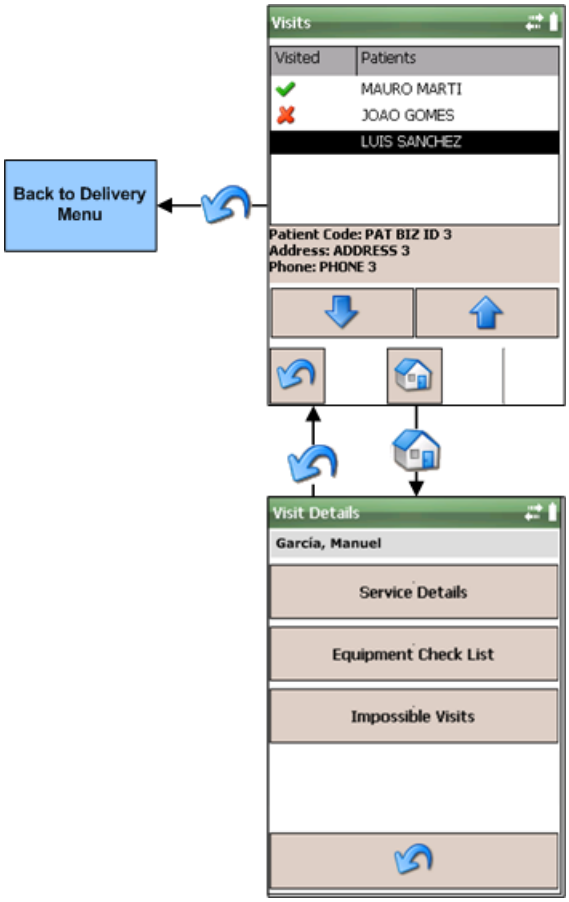

Figura 110: HMS3 Flujo detalles de visita

C.4.2. Detalles de servicio

Esta pantalla es la principal del apartado de visitas. Permite a los técnicos confirmar cada línea de detalle de la DN.

El contenido de la DN se muestra al técnico para que éste realice la operación correspondiente (entrega o recogida) del tipo de producto indicado para cada línea de detalle.

Normalmente cada operación del técnico corresponde con una línea de detalle de la DN, pero en algunos casos, el técnico debe comenzar una nueva línea de detalle.

Se muestra la siguiente información:

- Número de línea, recibido del servidor junto a la misma, o generado por el terminal si la línea es nueva.
- La cantidad esperada del producto a entregar o recoger.
- La cantidad real del producto que se ha entregado o recogido.
- La descripción del producto a entregar o recoger.

Figura 111: HMS3 Detalles de servicio

Reglas:

En este apartado el usuario puede:

- Editar una línea descargada del servidor.
- No se permite eliminar una línea descargada del servidor.
- Añadir una línea manualmente:
  - Editar la línea añadida.
  - Eliminar la línea añadida.
- Las líneas se muestran en el siguiente orden:
  - Líneas descargadas ordenadas por número de línea en la parte superior.
  - Líneas generadas por el usuario debajo de las descargadas.

C.4.3. Editar línea descargada

Mediante esta funcionalidad (), el usuario tiene la posibilidad de trabajar en una línea de detalle descargada del servidor.

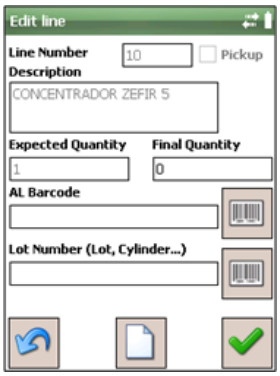

Figura 112: HMS3 Editar línea

Reglas:

- Cuando se edita una línea descargada del servidor, el usuario no podrá editar los siguientes campos:
  - Número de línea
  - Descripción de línea.
  - Cantidad esperada de la línea.
  - Indicador de recogida.

- Cuando la línea de detalle concierne a un dispositivo o cilindro, el técnico puede introducir el código de barras manualmente, o si está disponible, con el lector del terminal. Una vez que se ha capturado el código de barras, la cantidad final se iguala a 1 (no se trabajaba con líneas de detalle de este tipo de productos con cantidades superiores a uno). En todos los casos se debe validar el formato del código de barras, para ello el back office sigue una serie de reglas de negocio.

- Cuando la línea de detalle concierne a un consumible o cilindro, el técnico puede introducir el número de lote manualmente, o si está disponible, con el lector del terminal. En este caso no se validará el formato, y el usuario deberá especificar manualmente la cantidad final de productos recogidos o entregados.

- En caso de entrega, el código de barras o/y número de lote introducidos por el usuario se comparan con los disponibles en el stock del técnico: en el caso de que el producto no exista en el stock se espera la confirmación del usuario antes de validar la línea de detalle.

- Cuando se gestiona un servicio, el usuario solo podrá editar la cantidad final a 1,

para indicar que el servicio ha sido realizado (no se leerá el código de barras ni el número de lote).

- El botón reinicia la línea a su valor original.
- El botón valida cada línea.

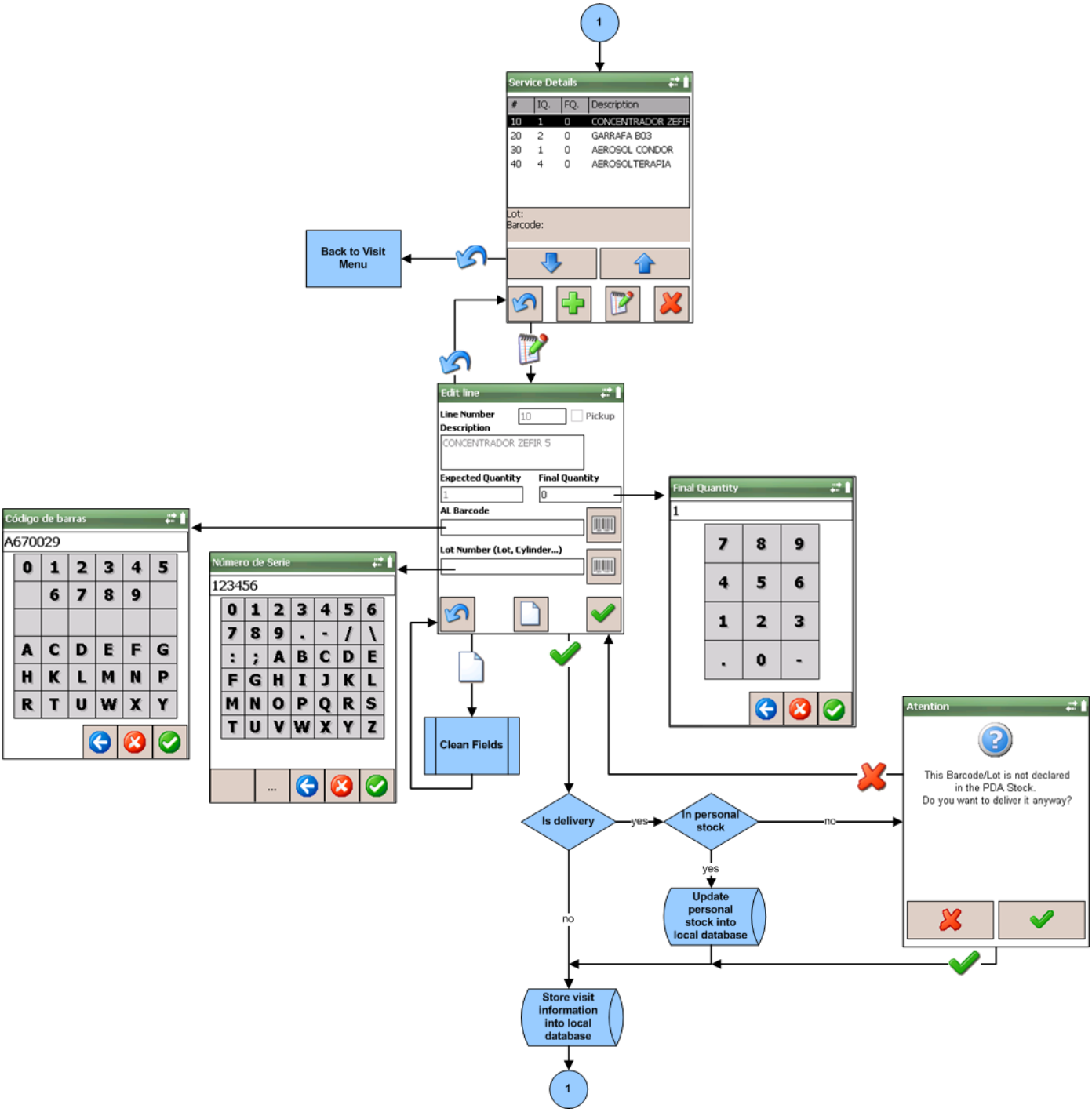

Figura 113: HMS3 Flujo edición de línea

C.4.4. Añadir una nueva línea

Esta pantalla permite al usuario añadir (+) nuevas líneas de tipo servicio, consumible o cilindro a una DN existente.

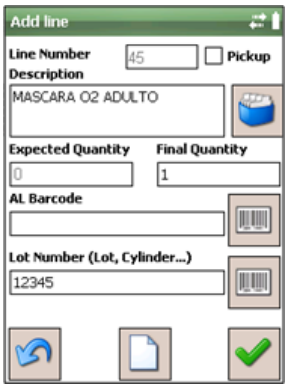

Figura 114: HMS3 Adición de línea

Reglas:

- Número de líneas: Al añadir una línea su número identificador debe ser igual al último número de línea del servidor más terminado en 5 (si tenemos descargadas las líneas 10, 20 y 30, y añadimos otra, la nueva tendrá como número de línea el 35).
- Recogida: El usuario debe indicar si la línea concierne a una recogida o a una entrega, mediante un indicador de recogida (checkbox).
- El resto de campos se completan del mismo modo que en la edición de línea.

C.4.4.1 Selección de producto

Al añadir una línea el usuario deberá seleccionar el tipo de producto entre cilindros, consumibles y servicios.

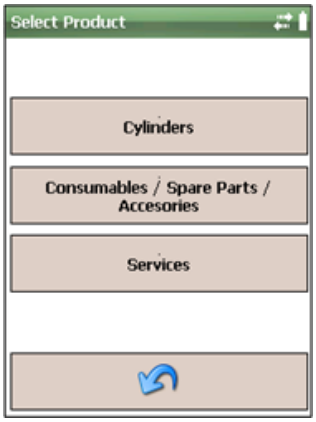

Figura 115: HMS3 Selección de producto

Reglas:

Tras la selección del tipo de producto se muestra la pantalla de selección correspondiente.

- Al seleccionar servicios se mostrará una lista con los servicios disponibles. Cuando se añade un servicio no se puede seleccionar el indicador de recogida.
- Al seleccionar cilindros se muestran los cilindros disponibles.
- Al seleccionar consumibles se muestran los disponibles con la cantidad restante en el stock del técnico. Una vez añadido, dicha cantidad decrece en uno.

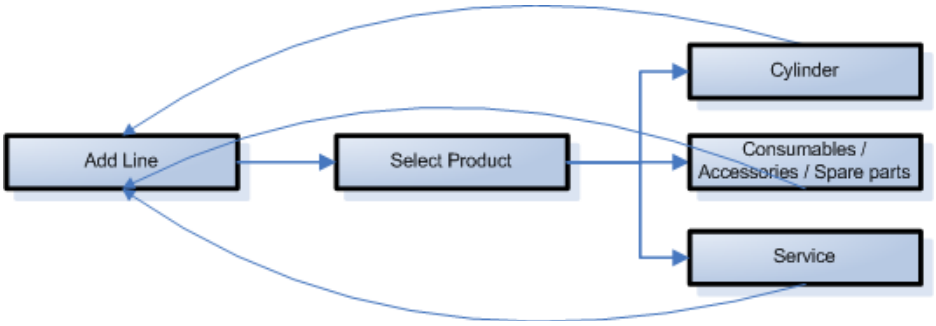

Figura 116: HMS3 Esquema selección de producto

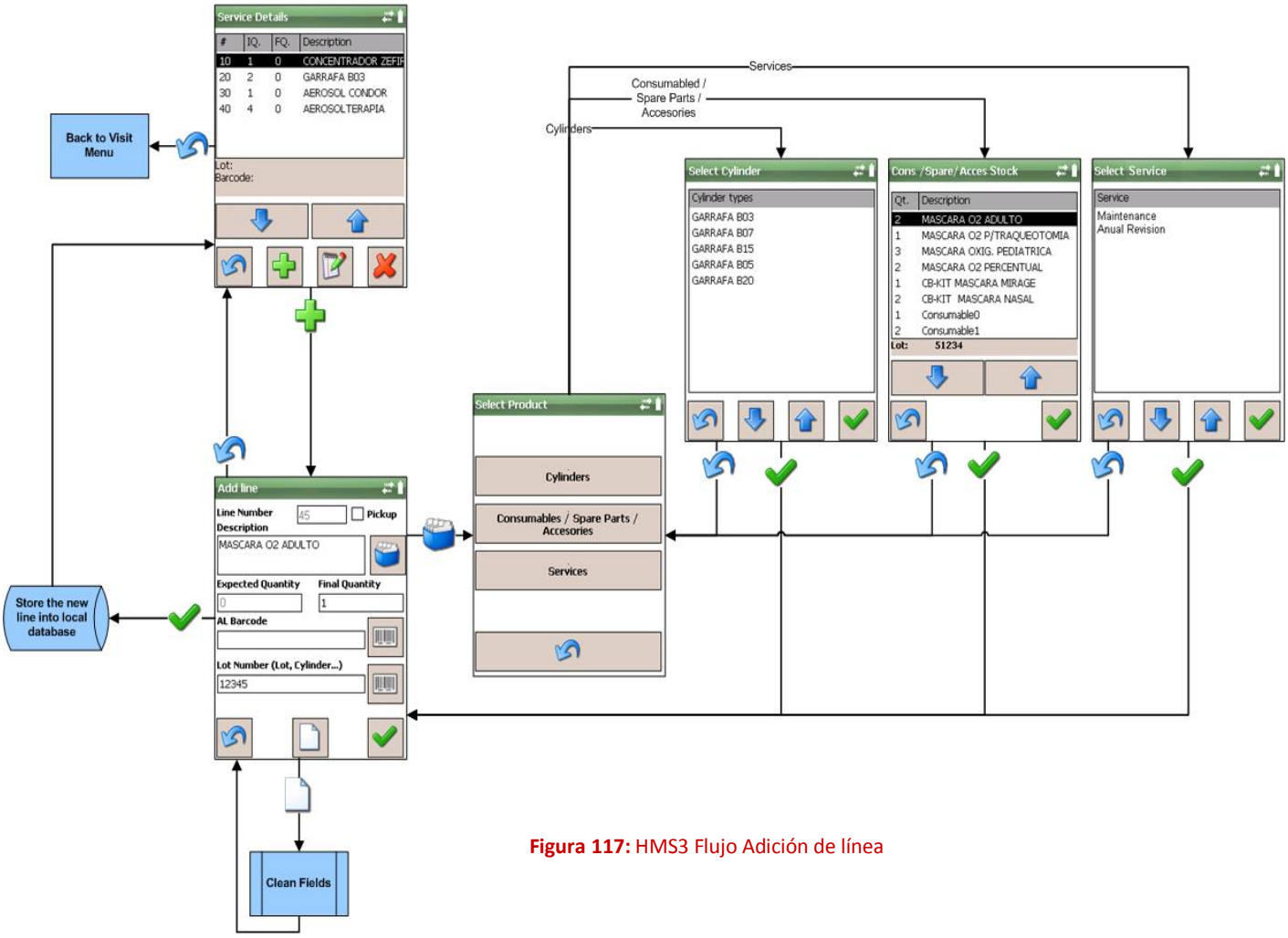

Figura 117: HMS3 Flujo Adición de línea

C.4.5. Borrar línea

El usuario podrá borrar una línea añadida por el usuario tras seleccionarla y pulsar el botón . No es posible borrar las líneas descargadas.

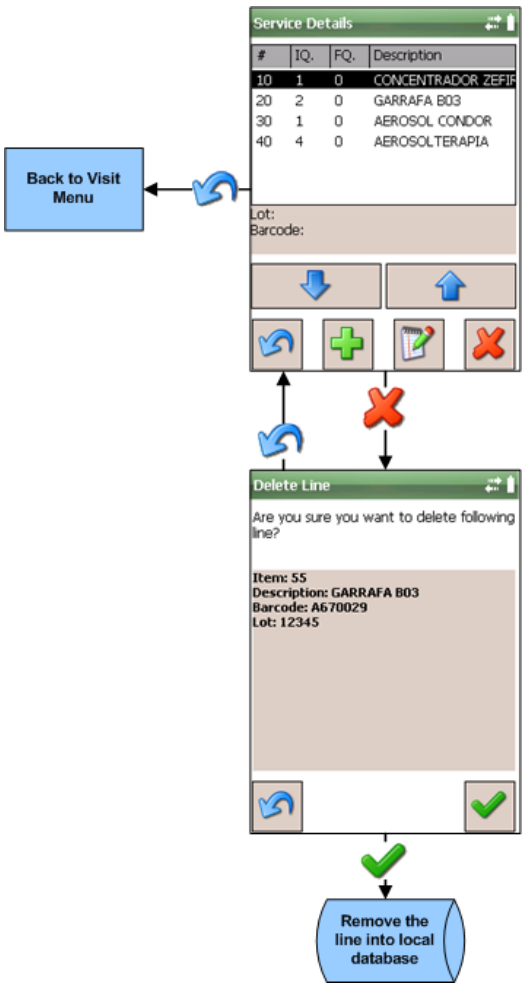

Figura 118: HMS3 Borrado de línea

C.4.6. Visita imposible

Es posible marcar la visita como imposible si el paciente está ausente, fallecido o ha sido imposible llegar al domicilio.

Figura 119: HMS3 Visita imposible

Reglas:

- No es posible marcar la visita como imposible si ha sido editada.
- Si una visita marcada como imposible es editada, debe volver a su estado inicial.

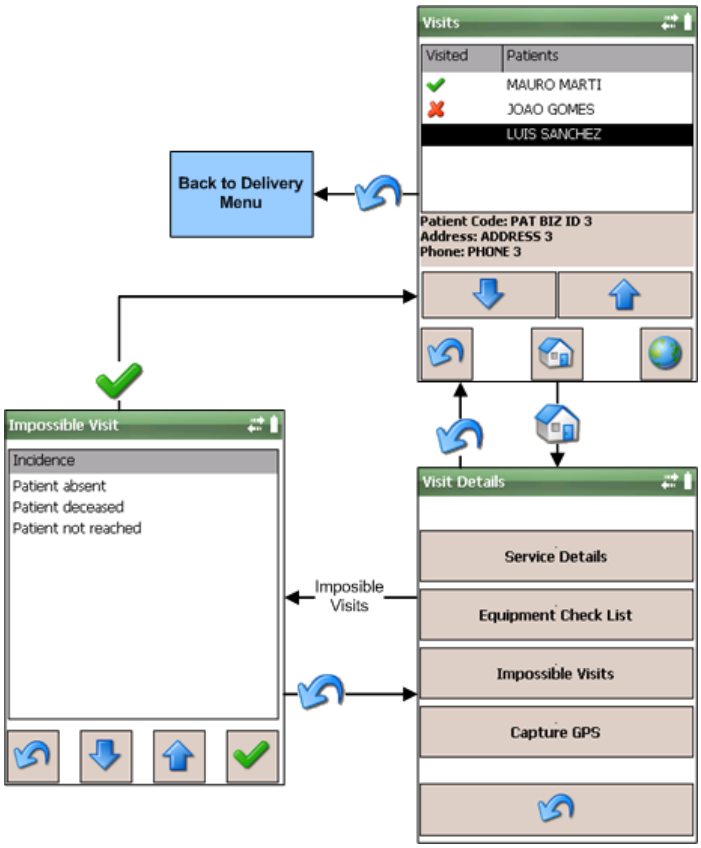

Figura 120: HMS 3 Flujo visita imposible

C.4.7. Cuestionario de dispositivo

El usuario podrá completar un cuestionario acerca de las operaciones que está realizando en el domicilio del paciente. El objetivo es completar cuestionarios específicos a un determinado dispositivo, acerca de las tareas a completar.

Los cuestionarios están almacenados en el servidor, existiendo diferentes modelos para cada dispositivo.

Reglas:

- El código de barras debe ser capturado con el lector o introducido manualmente.
- Al validar el cuestionario, y volver a esta pantalla, se muestran los valores validados.

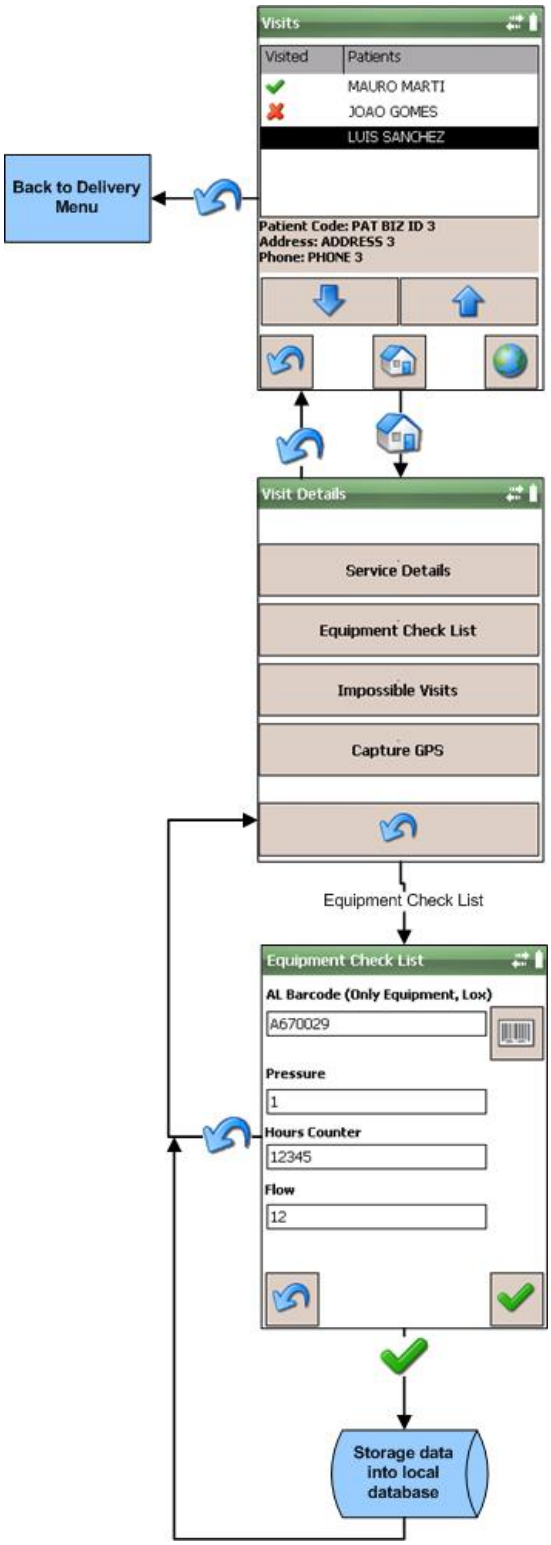

Figura 121: HMS3 Flujo cuestionario de dispositivo

C.4.8. Cerrar visita

En este paso el usuario debía cerrar la ruta, especificando la fecha y kilometraje final del vehículo. Aunque esta pantalla no desaparece en Homes.Android, la funcionalidad cambia completamente.

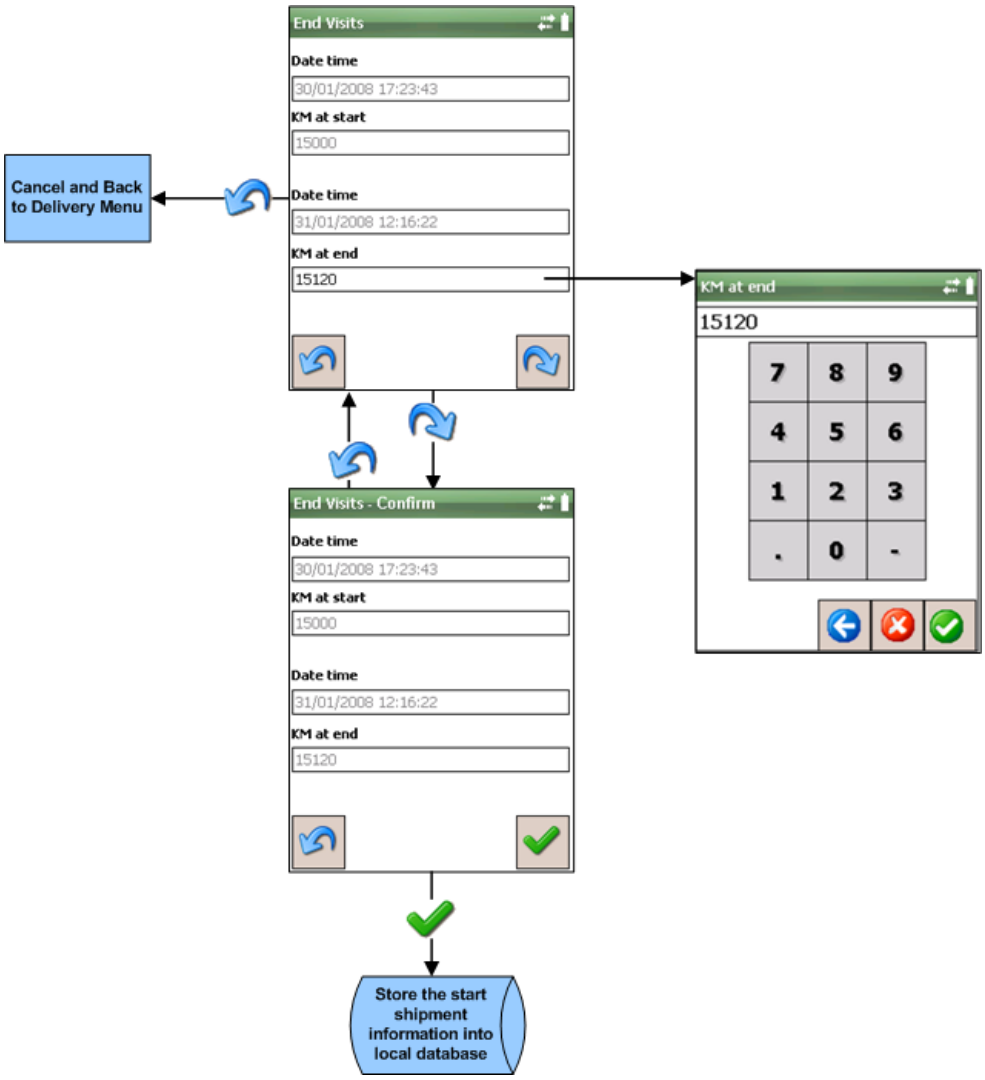

Figura 122: HMS3 Flujo cerrar visita

Anexo D: Análisis del Back Office

A continuación se definirán las diferentes capas del back office y su relación entre ellas, para ayudar a comprender los diagramas se presenta la siguiente leyenda.

Tabla 6: LEYENDA FIGURAS BACK OFFICE	
FIGURA	CAPA
- Circunferencia	Elementos de la capa tratada en el apartado
- Cuadrado	Elementos de otras capas

D.1. Capa de negocio

La capa de negocio gestiona la relación de la aplicación cliente con el resto de capas, para ello utiliza el patrón de diseño Facade [(23)].

Este patrón se basa en el uso de facades (fachadas), interfaces simples, a las que el sistema delega la taréa de seleccionar las subclases que se encargarán de realizar la acción requerida. Esto proporciona modularidad en sistemas complejos como éste, lo que simplifica el proceso de modificar o añadir nuevas funcionalidades, además de proporcionar trasparencia al back office.

El patrón funciona de acuerdo al siguiente esquema:

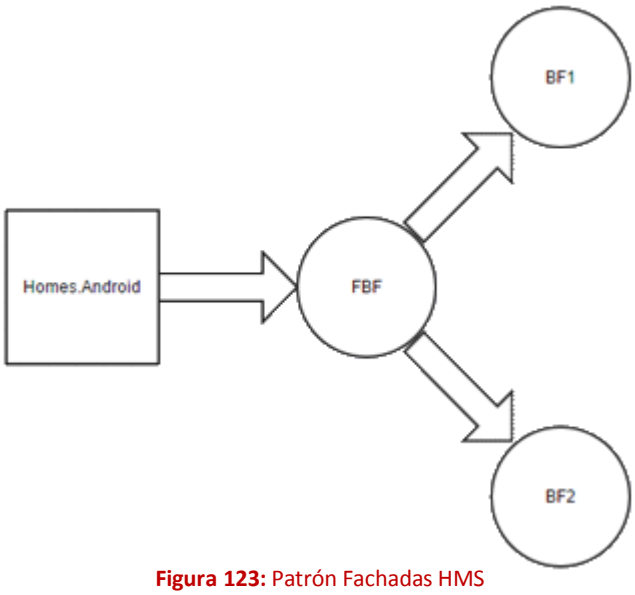

Figura 123: Patrón Fachadas HMS

Como se puede observar en la figura 23, el cliente (Homes.Android) delega la acción en la fachada FBF, que a su vez se encarga de seleccionar que subfachada se encargará de tratar esta acción.

A continuación se explicará más en detalle el funcionamiento de estas fachadas.

D.1.1. Functional Business Facades (FBFs)

Conjunto de fachadas mediante las que la aplicación realizará la comunicación con diferentes BFs

La aplicación cliente podrá acceder a cualquier FBF, pero nunca a un BF. De este modo los FBFs permiten una capa intermedia de acceso entre la aplicación y los FBFs.

D.1.2. Business Facades (BFs)

Subfachadas encargadas de la resolución de distintas operaciones, dichos módulos permiten una gran modularidad al encapsular las diferentes operaciones del sistema. Entre estos módulos se encuentran:

- Subfachadas de entidad: Existe un BF para cada par de objetos DAM DTO (véase D.3.2), desde el que se encapsulan las operaciones realizadas por el DAM y los posibles retornos en forma de objetos DTO a la aplicación.
- Subfachadas de comunicación con el servidor: Existe un BF encargado de las llamadas necesarias a los WebServices (HTTP Handlers) del sistema.
- Subfachadas de comunicación con la base de datos: Existe un BF específico para la creación y gestión de la base de datos local.
- Subfachadas auxiliares: Adicionalmente existen ciertos BFs encargados de otros tipos de operación, como la creación de identificadores únicos, formateadores de fecha o validadores de códigos de barra.

D.2. Capa de Datos

Esta capa gestiona los datos de la aplicación, a continuación se detallarán los contenidos de la capa.

D.2.1. CSV

Incluye una serie de clases destinadas a la creación y manejo de objetos CSV (Comma separated values) [24], estos objetos permiten la representación de tablas en documentos muy simples en los que los únicos delimitadores son la coma y el salto de línea.

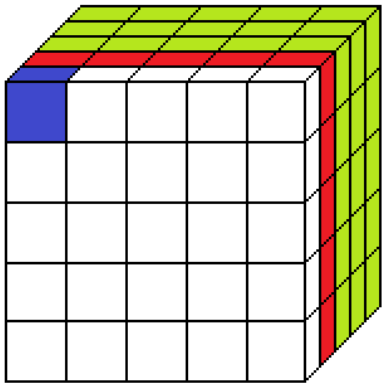

Figura 124: Estructura CSV

Se utilizan las siguientes clases:

- CSVFactory: Esta clase se encarga de la creación de los objetos CSV: CSVLine, CSVDoc y CSVCollection, además de incluir los campos separadores.
- CSVLine: Permite gestionar la creación de líneas CSV (Bloque azul en figura 23).
- CSVDoc: Permite gestionar la creación de documentos CSV compuestos por líneas CVS (Bloque rojo en figura 23).
- CSVCollection: Permite manejar una lista de documentos CSV y la lectura/escritura de los mismos en flujos de transmisión (Bloque verde en figura 23).

Las colecciones CSV juegan un papel fundamental en la transferencia de datos con el servidor. Debido a la restricción de ancho de banda del cliente, los datos enviados y recibidos entre el cliente y el servidor serán documentos CSV.

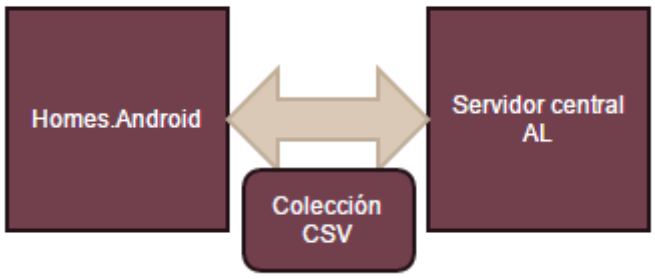

Figura 125: Esquema transferencia de datos

D.2.2. Data Access

Incluye una colección de clases orientadas al acceso a la base de datos, dichas clases se organizan de la siguiente manera:

- SqlBuilder: Clases que encapsulan las diferentes sentencias SQLite. Contiene una factoría encargada de la creación de las diferentes clases.
- Gestor de acceso a base de datos: Gestiona las operaciones base de las clases DAM, todas las clases DAM heredan de esta clase. Contienen una instancia del contexto de acceso a datos mediante el cual interactuarán con la base de datos. Contiene una instancia de las distintas sentencias SQLite obtenidas mediante la factoría de SQLBuilder.
- Provider: Interfaz que provee al contexto de los objetos de acceso a datos del framework .NET necesarios para la conexión con la base de datos. Las clases que implementan esta interfaz se ocupan de transformar las consultas a cada gestor de base de datos (SQLite/SQLServerMobile). Un buen ejemplo son las fechas, que requieren un distinto formateo para SQLite y para SQLServerMobile.
- Contexto de acceso a datos: Se encarga de gestionar la conexión con la base de datos para un contexto dado. Provee métodos para abrir y cerrar conexiones, comenzar y realizar transacciones, hacer rollback de transacciones fallidas, o crear comandos entre otros. Junto al contexto de acceso a datos se dispone de un gestor encargado de manejar los contextos de la aplicación, creando distintas instancias para los diferentes hilos de ejecución.

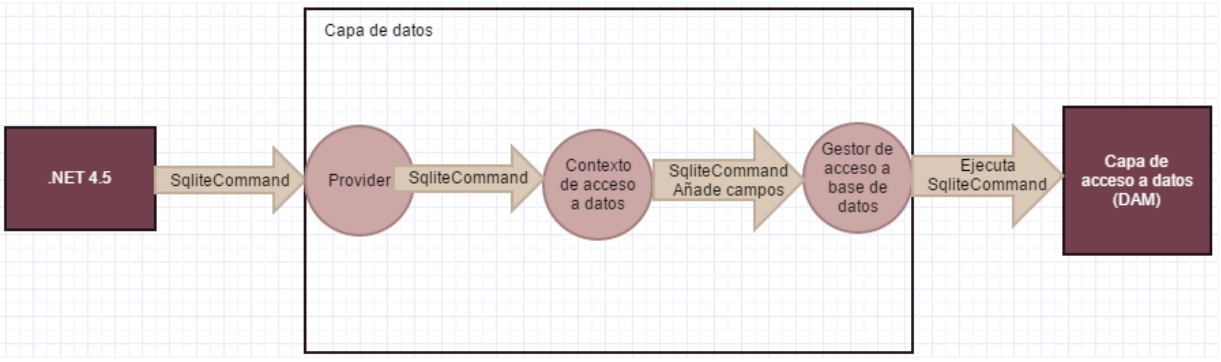

Figura 126: Diagrama capa de datos

D.2.3. SQLiteData

Proporciona una serie de utilidades para gestionar la creación de la base de datos por parte de la capa de acceso a datos (véase D.3).

D.3. Capa de acceso a datos

Esta capa sirve de puente entre la capa de datos y la aplicación cliente, permitiendo encapsular tanto los accesos a la base de datos (DAM), como los objetos de transmisión que utilizará la aplicación (DTO), a continuación se detallarán sus cuatro componentes:

D.3.1. Homes Data Context

Maneja el contexto de datos de la aplicación, este contexto proveerá a los BFs las factorías necesarias para la creación de los objetos DAM y DTO que veremos a continuación.

D.3.2. Data Transfer Object

Contiene los objetos de transferencia de datos que utilizará la aplicación para comunicarse con la base de datos, existe un DTO por cada tabla y un tipo de DTO especial, llamado CTO para objetos que se utilizarán para mostrar información (vistas con información de varias tablas). Estos últimos solo se utilizan dentro de la aplicación, y nunca se envían de vuelta a la base de datos debido a que no tienen una tabla asociada.

La aplicación recibirá estos objetos al hacer consultas a la base de datos mediante el DAM y del mismo modo los enviará con información al DAM cuando necesite enviar datos a la base de datos.

Todos los objetos de transmisión tienen dos métodos clave: ToCSV() y FromCSV(). Dichos métodos se encargan de la serialización y deserialización de información entre DTOs y CSVs.

D.3.3. Data Access Manager

Engloba los diferentes DAM (Data Access Manager) que utilizará la aplicación, cada uno de ellos encapsula las funciones necesarias para acceder a una determinada tabla de la base de datos, de este modo la aplicación podrá actualizar, insertar y hacer consultas a esta tabla mediante su DAM.

Al igual que sucede en los DTO, en este componente también se incluye un tipo especial de DAM solo utilizado para las vistas denominado CAM, este gestor especial encapsula las consultas a la base de datos que permiten la creación de CTOs.

Todos DAM heredan de la clase IDAM, e implementan los métodos abstractos de esta (update, updateAll…), sin embargo los CAM, debido a que no están asociados con una tabla real en la base de datos, no realizan ninguna acción en los métodos que impliquen añadir datos (insert, update…) controlándose mediante excepciones.

En el siguiente esquema se muestra el proceso de inserción de datos en la base de datos local mediante el DAM.

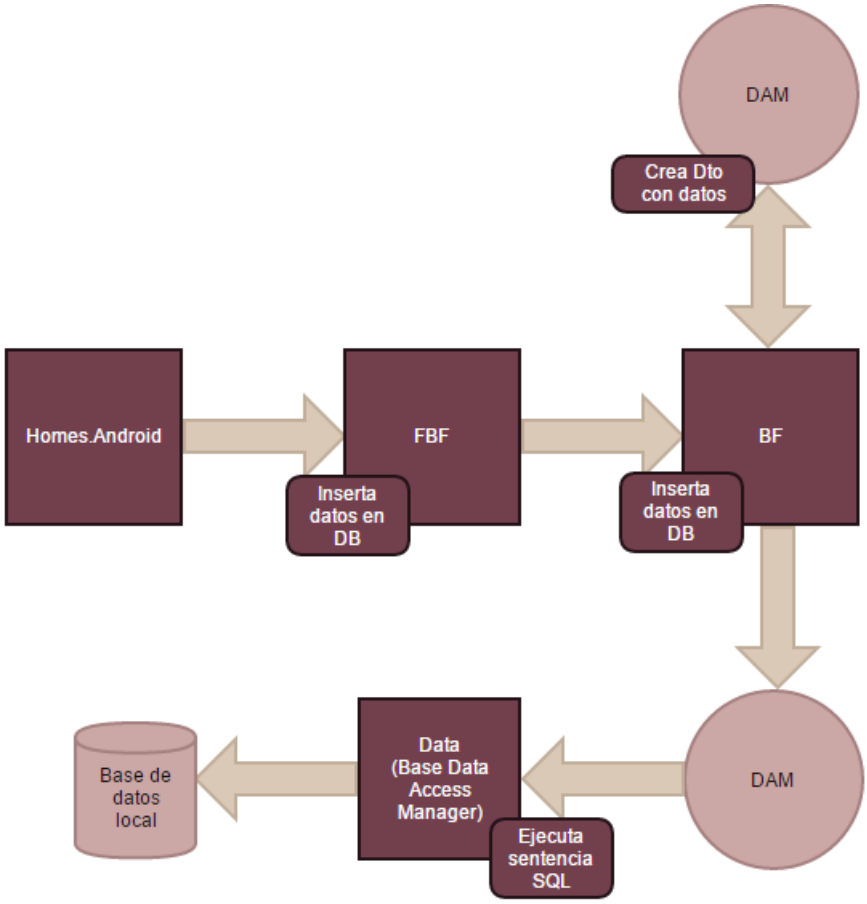

Figura 127: Diagrama acceso a base de datos

Para recuperar datos, el proceso es muy similar, creando un objeto DTO vacío del tipo que se esté tratando de recuperar en el correspondiente BF, y asignando este objeto al retorno del método del DAM que hará la consulta en base de datos. Posteriormente se irá devolviendo este objeto hasta que llegue a la aplicación cliente.

D.3.4. DataBaseManager

Esta clase se encarga de la creación y mantenimiento de la base de datos. Permite comprobar si la base de datos existe, cerrar la conexión, cargar esquemas, y demás operaciones sobre la base de datos. Para ello se sirve de las clases de SQLiteData (véase D.2.3).

D.4. Capa de manejo HTTP

Esta capa es heredada de las anteriores versiones de HMS y no se ha modificado. En la siguiente ilustración se muestra el proceso de recuperación de datos del servidor central.

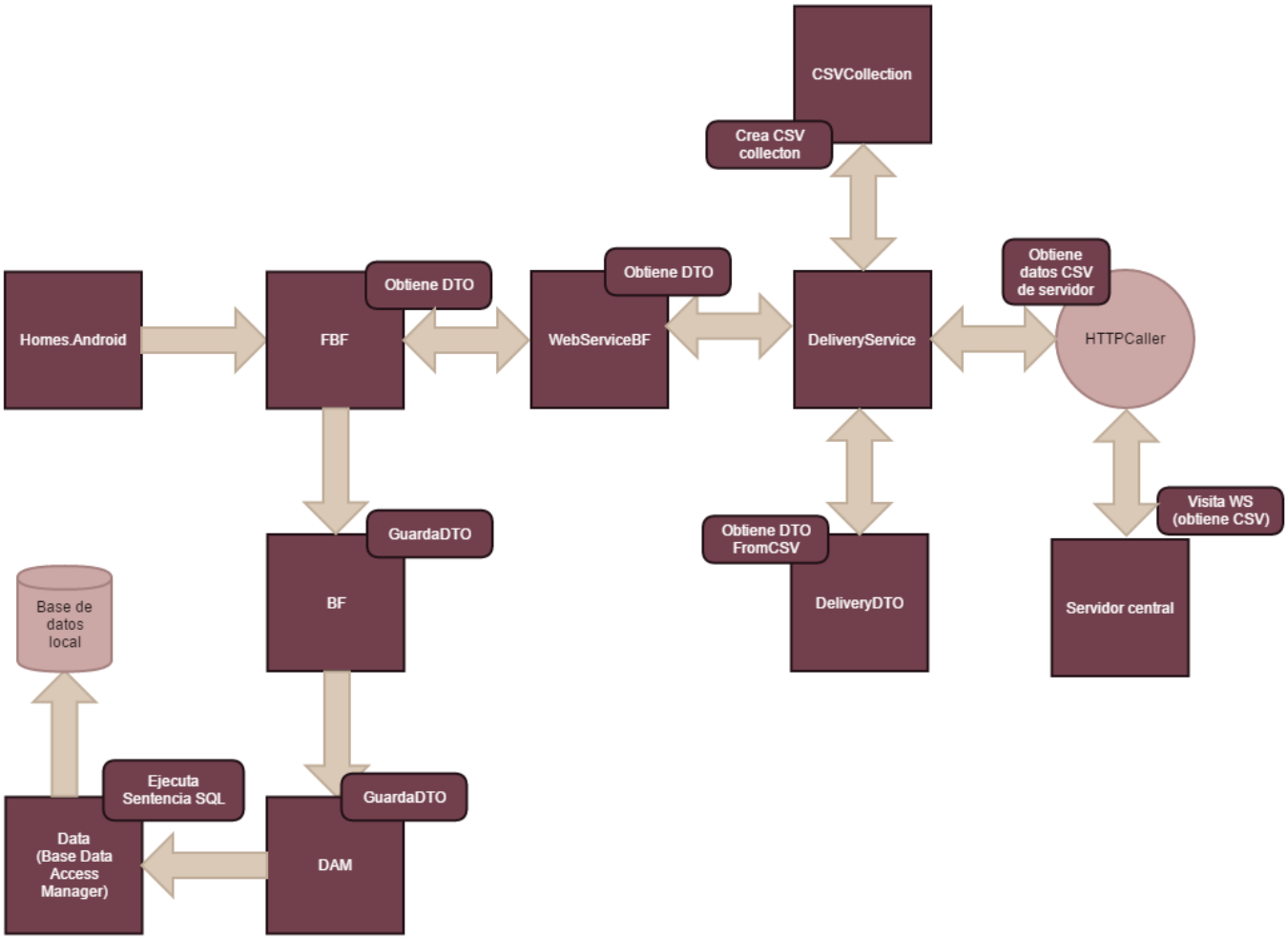

Figura 128: Diagrama llamada HTTP

Anexo E: Manejo de iTextSharp

En este apartado se documenta el funcionamiento [25] y usos dados de la herramienta iTextSharp dentro de la aplicación.

Esta herramienta consiste en una portabilidad de iText para C# y permite generar PDFs mediante código. Para gestionar la generación de documentos PDF con esta herramienta hay que seguir los siguientes pasos:

- Creación del documento:
  - Crear un Filestream donde se creará el fichero.
  - Instanciar un nuevo objeto del tipo Documento de iTextSharp.
  - Instanciar un nuevo objeto del tipo writer de iTextSharp, este objeto se encargará de la escritura de campos en el documento.
  - Abrir el documento e incluir el contenido:
    - Para manejar el contenido, esta herramienta permite el uso de tablas. La ventaja de esto es la posibilidad de crear tablas dentro de tablas, lo que permite una gran flexibilidad a la hora de organizar el contenido de los documentos.
    - Existe la posibilidad de modificar bordes y márgenes de columnas, además de combinar tanto filas como columnas.
  - Cerrar el documento.

- Añadir firma con certificado:
  - Crear un filestream donde se creará el fichero firmado.
  - Instanciar un nuevo objeto del tipo reader de iTextSharp, este objeto se encargará de la lectura del documento previamente creado.
  - Instanciar un nuevo objeto del tipo stamper de iTextSharp, este objeto se encargará del estampado de la firma en el documento.

- Añadir numeración:
  - Instanciar un nuevo objeto del tipo PdfTemplate de iTextSharp e incluirlo en la columna donde se espera la numeración. Este objeto se escribe en el documento al cerrar el mismo, permitiendo mostrar el número de hojas desconocido hasta el cierre [26].
  - Implementar la interfaz IPdfPageEvent de iTextSharp, que permite capturar diferentes eventos relacionados con la creación del PDF.

- Dentro del método que captura el instante en el que el documento se cierra, sustituir el template por el número de hojas del documento.

➢ Añadir cabecera:

- Dentro del método de la interfaz IPdfPageEvent que captura el instante en el que el documento crea una página, añadir la tabla con la información de la cabecera al documento.

Anexo F: Abordando el manejo del lector lineal

Se documenta el análisis y elección del procedimiento más adecuado para tratar el lector lineal del terminal Motorola TC55.

F.1. Implementación mediante Intent y Datawedge

Por defecto, todos los terminales TC55 traen instalada la aplicación DataWedge encargada del manejo del escáner lineal, esta aplicación permite la gestión de diferentes perfiles.

Es posible configurar un perfil de DataWedge, que envíe intents con los datos recibidos a una determinada aplicación [(27)].

Configurando un perfil para HOMES, y preparando la aplicación para recibir intents externos (especificando los permisos en el manifiesto) se puede sobrescribir el método OnNewIntent [(28)] para recibir los datos de DataWedge.

De este modo podríamos capturar códigos de barras desde la aplicación lanzando el escáner tanto desde la interfaz de la aplicación (mediante un intent a DataWedge) o mediante la interfaz física (el botón físico, asociado también al perfil de Datawedge), recibiendo el resultado en ambos casos mediante el intent lanzado por el DataWedge.

Esta implementación es la más simple, pero presenta dos problemas: La configuración del perfil de DataWedge en todos los terminales y la posibilidad de que estos perfiles sean modificados por el usuario alterando así la funcionalidad del lector. Debido a esto, se desestima el uso de esta alternativa.

F.2. Implementación mediante librería EMDK

Motorola proporciona una librería JAR [(29)] con las clases necesarias para el manejo del

láser del terminal.

Para utilizar librerías de Java en Xamarin es necesario enlazarlas, para este propósito Xamarin proporciona una guía en su web [(30)].

En la guía se explica cómo crear una librería DLL de Microsoft mediante la plantilla de enlazado disponible en Xamarin.

En la web de Motorola, se avisa sin embargo, que para enlazar su librería, será necesario realizar ciertos cambios respecto a la guía de Xamarin [(31)].

Aunque la plantilla de enlazado de Xamarin transforma la librería automáticamente, se permite modificar esta transformación modificando el archivo metadata.xml [(32)].

Al realizar la transformación automática, ciertos campos de la biblioteca original desaparecen. Esto sucede porque el enlazador automático no es sensible a las minúsculas e ignora cualquier campo con el mismo nombre.

Para solucionar esto es necesario editar el fichero metadata.xml, con la siguiente sentencia, que permite renombrar el campo problemático:

```
 <attr
path="/api/package[@name='com.symbol.emdk.barcode']/class[@name='Scanner']/field[@name='triggerType']"
name="managedName">NewTriggerType</attr>
```

Una vez adaptada, la librería permite más control sobre el escáner respecto de la solución anterior, sin embargo también presenta una desventaja.

La biblioteca necesita que el escáner se maneje de distinta forma si se activa desde la aplicación o desde el botón físico, siendo una funcionalidad exclusiva de la otra, y no proporciona una funcionalidad robusta para el intercambio entre un modo y otro en tiempo de ejecución (permite el cambio, pero no es fiable).

La solución implementada consiste en capturar el pulsado del botón hardware desde la aplicación, y lanzando el escáner desde la misma.

Al implementar esta solución, es importante deshabilitar el escáner antes de salir o matar la aplicación, ya que al tomar el software el control del escáner, no permite que éste se lance en ninguna otra aplicación simultáneamente.